THE GEORGE WASHINGTON UNIVERSITY
SCHOOL OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

COLLISION AVOIDANCE SYSTEM FOR THE VISUALLY IMPAIRED

Final Product Review

By

Brandon Bernier and Srinivas Tapa

Prepared for:
Professor C. E. Korman
ECE 4925W Section 10:
ECBE Capstone Project Laboratory III

Edited: May 12, 2014

Abstract

The Collision Avoidance System assists the visually impaired while they walk by alerting them to obstacles in their path. It is capable of detecting obstacles at both head-level within 5 feet and ground-level within 7 feet and alerting the user with automatic audio alerts. The system consists of a peripheral device to detect head-level objects, another to detect obstacles on the ground, and an Android™ smartphone application to connect the system and generate the audio alerts. Two peripheral devices with integrated ultrasonic sensors and accelerometers are used to detect obstacles and prevent collisions while the user walks. A glasses-like headset ensures that the user is protected from collisions at head-level and an additional small attachment placed on the user's walking cane senses obstacles on the ground. Both of these devices communicate with the smartphone via Bluetooth® wireless technology. Additionally, the phone uses built-in text-to-speech functionality to generate audio alerts that can be easily understood by the user. Combined, the system protects the visually impaired from obstacles that are otherwise a daily challenge and danger. This document describes the entire design process for the Collision Avoidance System including all major design decisions, testing procedures, preliminary test results, as well as an explanation of the breakdown of work between all team members.

Table of Contents

List of Figures

List of Equations

List of Tables

List of Datasheets

1. Introduction

1.1. Background

Vision is the ability to interpret the surrounding environment by processing information that is contained in visible light. Loss of vision, or blindness, can be caused by a variety of physiological or neurological factors such as cataracts, glaucoma, age-related macular degeneration, childhood blindness, and many other causes. The World Health Organization (WHO) estimates that 285 million people worldwide are visually impaired, 39 million of which are completely blind.[1] Cures, treatments, and medical devices to deal with blindness have been evolving through the years, resulting in Braille, retinal implants and transplants, and specialized walking canes. Yet, for the permanently blind, the range of options is very limited. Most employ the use of aids such as nurses and seeing-eye dogs in addition to walking canes, but new research is striving toward integration between the brain and optical devices. For example, researchers James Weiland and Mark Humayun are working to develop artificial retinal technology, which consists of microelectrodes implanted within the eye that receive Laser or RF transmitted from a camera that then activates neural cells.[2] The proposed Collision Avoidance System would allow a blind individual to navigate their environment with relative ease by alerting them to obstacles in their immediate pathway.

The main purpose of the system is to automatically measure the distances of objects within a predefined range, process this information through an Android™ cellular device, and then relay the information to the user in feet and inches. The device will consist of a headset containing sensors that will be able to detect objects at head-level of the user. Additionally, a walking cane attachment will be designed to detect ground-level obstacles. This attachment will contain sensors in combination with an accelerometer to gauge the distance of objects in front of the user. All information is communicated wirelessly using Bluetooth between the external devices and the smartphone, which then relays this information as spoken audio alerts. The figure below shows an example of a walking cane for the visually impaired as well as a basic illustration of the system.

Figure 1.1: Example of walking cane and basic system implementation.[3]

1.2. Market Justification

Census data collected nationwide in 2011 found that around 7 million people in the United States have some type of serious visual disability. Furthermore, of the 7 million individuals suffering from serious visual disabilities that are also between the ages of 21 to 64, only 23.8% of these individuals are employed full-time.[4] The total economic impact of vision loss in the United States is estimated at nearly $68 billion annually.[5] The market for assistive technology for those with visual impairments is vast.

The consumer market for the visually impaired is filled with a wide range of products that claim to provide major advantages to the daily lives of the visually impaired. Such products include laser guided canes, sonic devices, and GPS guidance systems. A device similar to the proposed system is iGlasses™ by AmbuTech Inc., which is an independent pair of glasses that detect objects in front of the user's head.[6] The device alerts users by vibration in the glasses to objects within its 3 meter range and has a max battery life of 10 hours of continuous use. This system retails at $96.10, while the proposed system would retail at roughly $664, however, with much more functionality.

The proposed Collision Avoidance System goes beyond this in that it detects obstacles on the ground as well and would have much longer battery life. In addition, the interface with an Android cellular device will allow for the system to output audio feedback regarding objects in terms of distance from the user. Due to the number of blind individuals in the United States as well the many unique qualities of this system in comparison to competitive products, the proposed Collision Avoidance System is a marketable product.

2. Overall System Requirements and Specifications

2.1. Overall System Requirements

- The system should assist in avoiding collisions while a user walks
- The system should detect obstacles at ground-level in the user's path
- The system should detect obstacles at head-level in the user's path
- The phone application should run on a smartphone using the Android platform
- Both peripheral devices should be powered by separate rechargeable batteries
    - Batteries should last the user an entire day
- Both peripheral devices should be able to be turned on and off
- Both peripheral devices should wirelessly communicate with the smartphone
- Both peripheral devices should be small and light enough so as not to impede normal motion
- The system should give the user spoken audio alerts and notifications
    - Should alert user to obstacles on the ground with distance to the object
    - Should alert user to head-level obstacles with distance to the object
    - Should alert user once an obstacle is gone and their path is clear
    - Should alert user when the cane is improperly oriented
    - Should alert user when the battery in either peripheral device is low
    - Obstacle alerts should be triggered in time for the user to react
- The system should be designed to conform to engineering standards pertaining to telecommunications and information exchange between systems
- The system should be designed to conform to engineering standards pertaining to SOIC package specifications
- The system should be designed to conform to engineering standards pertaining to Micro Universal Serial Bus specifications
- The system should be designed to conform to engineering standards pertaining to terms, definitions, and letter symbols for microelectronic devices

2.2. *Overall System Specifications*

- Horizontal object detection: ±20˚; Vertical object detection: ±10˚
- Ground-level obstacle detection range: 0-7ft ± 0.5ft
- Head-level obstacle detection range: 0-5ft ± 1in
- Smartphone Application Compatibility: Android 4.2
- Peripheral devices will be powered by independent, rechargeable batteries
  - 3.7V 1000mAh Li-Ion rechargeable battery
  - Batteries will be recharged via standard micro-B USB connector
  - Use time per full charge: > 24 hours
- Will have physical switches on both peripheral devices to turn them on and off
- Peripheral devices will use Bluetooth 2.1
- Headset Weight: < 10oz; Form factor: glasses with < 5in^3 housing for circuitry
- Cane Attachment Weight: <10oz; Size: < 6in^3
- The system will give the user audio alerts and notifications
  - Will alert user to ground-level objects with "*Ground object X feet ahead*" audio output and "*Ground Clear*" once the path is clear
  - Will alert user to head-level obstacles with "*Head obstacle X feet ahead*" audio output and "*Head Clear*" once the path is clear
  - Will alert user with "*Rotate cane clockwise* or *counter-clockwise*" audio output
  - Will alert user "*Headset* or *Cane, battery low*" when battery is below 20%
  - Response time: < 100ms
- The system will be designed to conform to engineering standard IEEE 802.15.1™-2005 13 Security[17]
- The system will be designed to conform to engineering standard MS-013 from JEDEC Solid State Product Outlines[18]
- The system will be designed to conform to engineering standard Universal Serial Bus 3.1 Specification, Revision 1.0[19]
- The system will be designed to conform to engineering standard JESD99C Terms, Definitions and Letter Symbols for Microelectronic Devices[20]

3. Approach to Overall Design

3.1. Current Design

The Collision Avoidance System receives information about the user's immediate environment including both head-level and ground-level obstacles in front of the user and provides audio alerts for the user when a collision is possible.

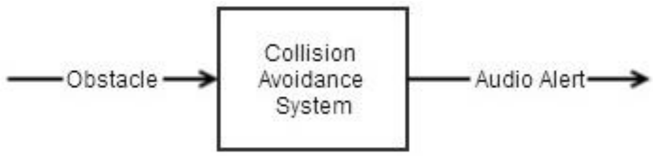


Figure 3.1: Context Diagram of entire system.

To satisfy the requirements of the project, it is necessary to have multiple sub-systems capable of interacting and performing the various required tasks. There must be sensors capable of detecting oncoming obstacles, sensors to determine the orientation of the user's head and cane, microcontrollers to interpret the data from the sensors, a smartphone capable of integrating multiple peripheral devices and generating audio alerts, and a means of communication between all sub-systems of the design.

Because the system must protect against obstacles not only at head-level but also on the ground, two sub-systems have been designed to accomplish this task. The first is a glasses-like headset and the second is a small attachment mounted towards the bottom of a user's walking cane. Initially, only a single, head-mounted device was envisioned; however, as the design evolved, it was decided that a secondary sub-system attached to the user's cane would best meet the requirements of the design and provide significant additional value to the user. Keeping the headset as light and small as possible is important to the overall design, so splitting off the ground-level obstacle sensing to a separate sub-system made the most practical sense.

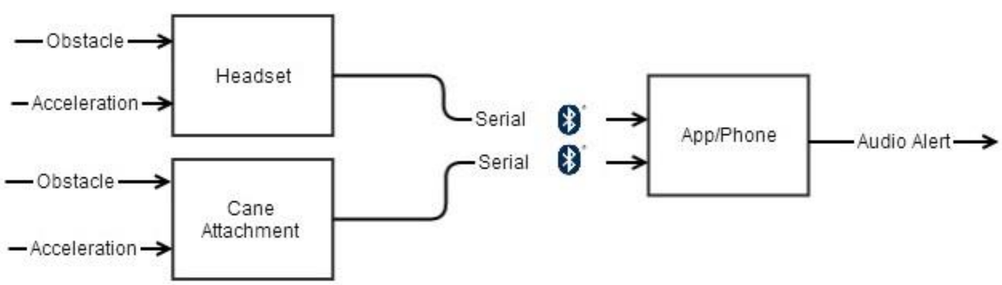


Figure 3.2: Data Flow Diagram showing individual sub-systems.

In order to sense the distance to upcoming obstacles, a set of sensors must be used. For this system, ultrasonic sensors were chosen for their accuracy at both short and long ranges as well as their relatively compact size.

In addition, accelerometers will be used to determine the orientation of both the user's head and cane. Accelerometers are capable of sensing acceleration, including the constant acceleration due to gravity ($g=9.81m/s^2$). By measuring the change in g's on a specific axis, it is possible to determine the angle at which the accelerometer is oriented with respect to a predefined zero point. This is beneficial for the headset because it is necessary to ignore irrelevant objects that may be detected by the ultrasonic sensor that are not actually in the user's path. For instance, if the user looks up, the system should not mistakenly think that the ceiling is an obstacle in the user's path. Similarly, an accelerometer will be useful on the cane attachment to ensure that the user always keeps the device properly oriented on top of the cane, facing forward.

Once the sensors detect objects, a microcontroller must be used to interpret and process the data. The microcontroller will receive input from the sensors and generate an output that can be sent to the next module within each sub-system. Microcontrollers act as a central hub capable of controlling both the ultrasonic sensors and the accelerometers simultaneously, then transmitting data to a Bluetooth transceiver.

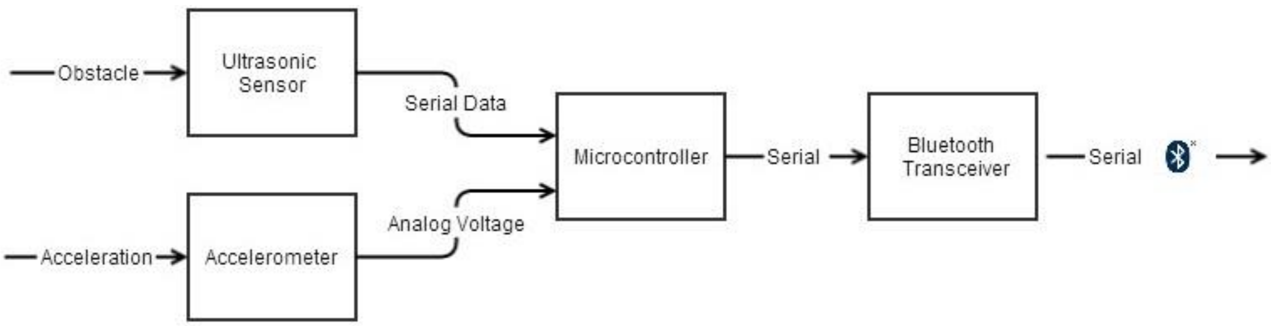


Figure 3.3: Data Flow Diagram for both Headset & Cane Attachment sub-systems.

Transmission of data between the two peripheral sub-systems and the smartphone needs to be both reliable and done without hindering the overall product. For this reason, it was determined that the devices would be wireless and communicate with the phone using a common wireless technology such as Bluetooth.

Once the smartphone receives the data from the two peripheral sub-systems, it must be programmed to interpret and use that data to generate audio alerts for the user. The smartphone will utilize built-in functionality of the Android platform to generate spoken alerts for the visually impaired user to notify them that there is an obstacle at a given distance away. Limits such as how often alerts can be triggered will be crucial in making the system not only functional but also practical to use.

Figure 3.4: Data Flow Diagram for Android application sub-system.

Together, these individual sub-systems and specific modules will be capable of meeting the defined general requirements and specifications for the system. The following tree diagram shows the breakdown of the overall system into sub-systems, modules, and sub-modules. Note that the headset and cane attachment sub-systems consist of identical modules and have only been represented once.

Sub-System	Inputs	Outputs
Headset	Obstacle, Acceleration	Serial Bluetooth Signal
Cane Attachment	Obstacle, Acceleration	Serial Bluetooth Signal
Android Application	Serial Data from Peripherals	Audio Alerts

Table 3.1: Sub-system inputs and outputs.

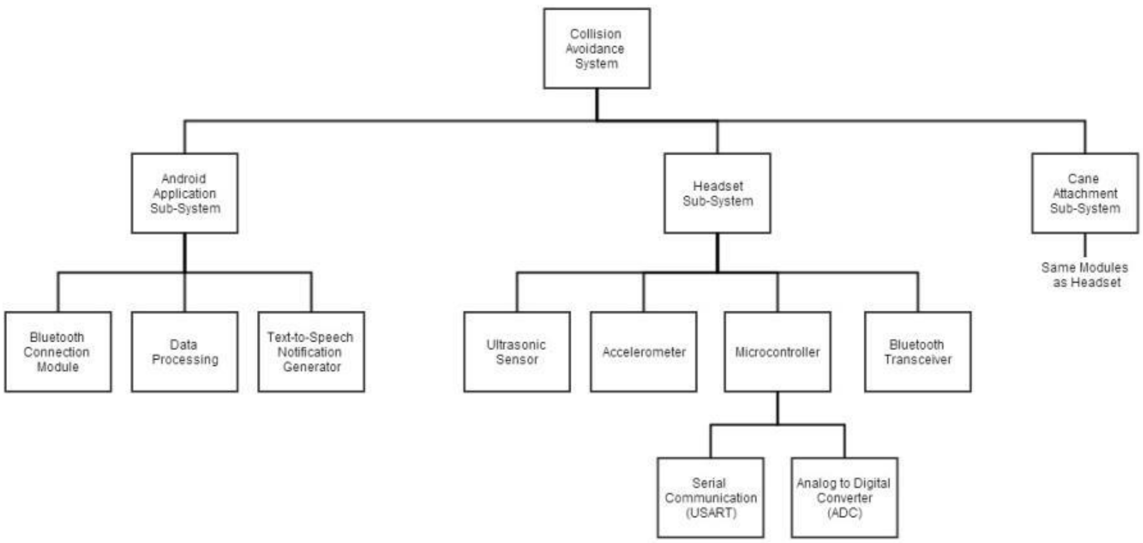


Figure 3.5: Tree Diagram of entire system.

3.2. Evolution of the Current Design

In designing the Collision Avoidance System for the visually impaired, certain critical engineering challenges were encountered and needed to be solved. The most critical specification for the system is that the response time will be less than 100ms, which was found using **Equation 1**. If walking at 5ft/second[7], this equates to notifying the user within 0.5ft. This estimate is conservative in that average walking speed tends to vary between 3-5ft/second.

$$t = \frac{d}{v} = \frac{response\ distance}{walking\ velocity}$$
$$t = \frac{0.5ft}{5ft/s} = 100ms$$

Equation 1: Calculation for desired response time.

Another challenge that needed to be solved was the 24-hour operation requirement. The proposed ultrasonic sensor, accelerometer, microcontroller, and Bluetooth transceiver draw roughly 2mA[8], 400µA[9], 2mA[10], and 30mA[11], respectively, while operating. The resulting overall current draw of about 35mA must be used to calculate the necessary battery capacity as shown in **Equation 2**. The batteries for the peripheral devices also need to be low voltage for the embedded components, yet fairly high capacity to power the devices for 24 hours of continuous use.

$$
\begin{aligned}
&Battery\ Capacity = current\ draw * hours \\
&Battery\ Capacity = (\sim 35mA)(24\ hours) \\
&Battery\ Capacity \approx 840mAh
\end{aligned}
$$

Equation 2: Calculation for minimum battery capacity.

When it comes to designing the housing of both peripheral devices, materials such as aluminum and plastic were considered for both their durability and lightweight characteristics. For this project, the housing is made of 3D-printed plastic because it is easy to have produced rapidly and at a low cost, while also being extremely lightweight. Additionally, the housing for the cane attachment must be compact enough to sit on the cane and must be designed to attach securely on the cane. A rubberized Velcro strap will be attached to the housing to secure it to the user's walking cane. Each housing has two cutouts for the micro-B USB connector and an on/off switch. The headset housing also has a small hole to allow for wires to go out to the ultrasonic sensor, which will be mounted separately on the headset. **Equation 3** shows a preliminary calculation for the estimated size of the cane attachment housing.

$$
\begin{aligned}
&V = (lwh)_{back} + (lwh)_{front} \\
&V = (2in)(1.8in)(1in) + (1in)(1.8in)(1.5in) \\
&V = 6.9in^3
\end{aligned}
$$

Equation 3: Calculation for cane attachment volume.

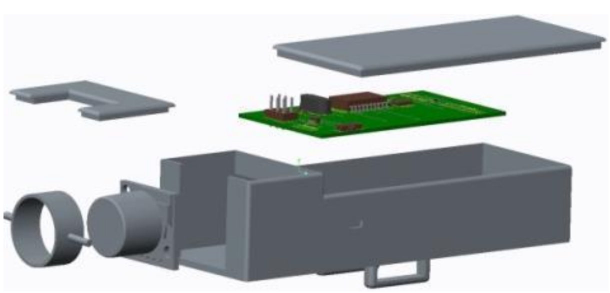


Figure 3.6: Cane attachment 3D view.

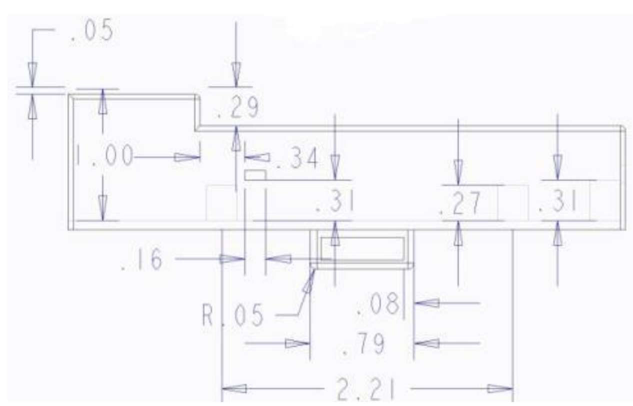


Figure 3.7: Dimensional drawing of cane attachment.
Note: Units in inches for all dimensional drawings.

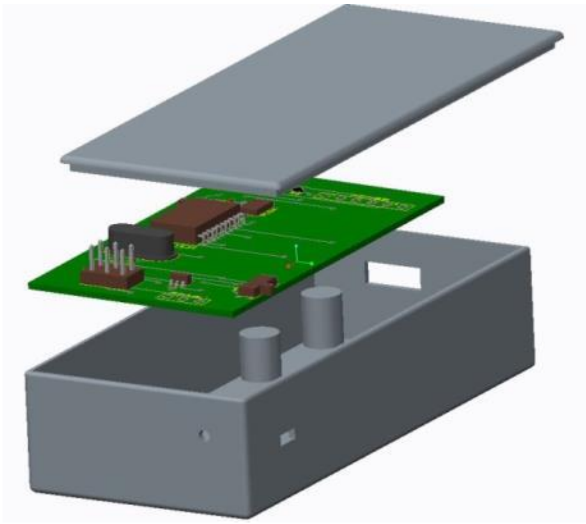


Figure 3.8: Headset housing 3D view.

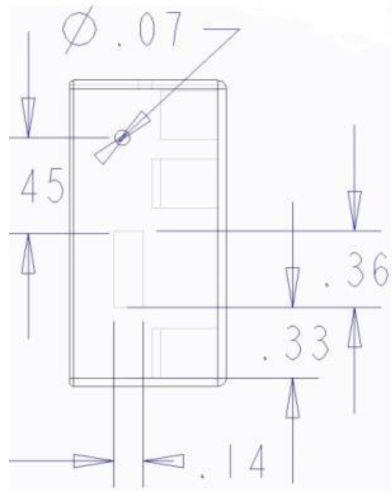


Figure 3.9: Headset housing dimensional drawing.

$$V = lwh$$
$$V = (2.5in)(1.8in)(0.69in)$$
$$V = 3.11in^3$$

Equation 4: Calculation for headset attachment volume.

One key technical challenge to overcome was designing the cane attachment so that the ultrasonic sensor always faces forward parallel to the ground. One solution would be to simply add more sensors so that at least one always points in the right direction; however, this idea would fail to meet the size requirements for the device. The final decision was to include a gimbal-based solution that allows the ultrasonic sensor to rotate freely along a single axis. The gimbal itself will be made of miniature ball bearings mounted inside the housing and the ultrasonic sensor will be modified to have small pins that can fit into the ball bearings. These pins must be placed at the sensor's center of gravity so that while level, the sensor sits parallel to the ground.

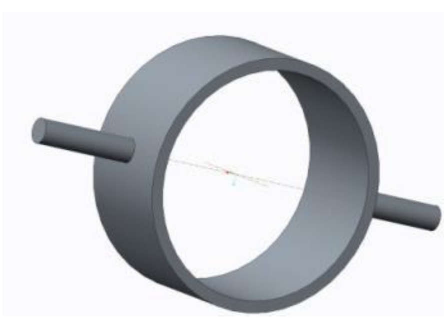


Figure 3.10: Sensor casing 3D view.

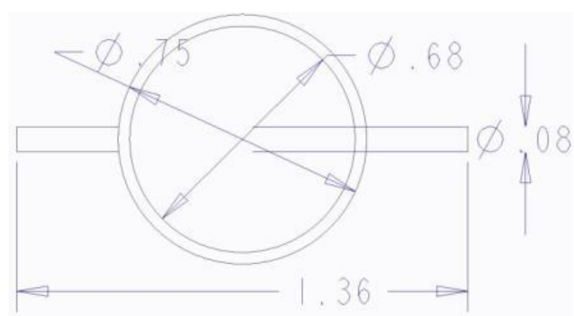


Figure 3.11: Sensor casing dimensional drawing.

Additional dimensional drawings can be found in ***Appendix A: Mechanical Drawings*** including full three-dimensional views and measurements for the various housing components.

3.3. *Applicable Standards If Commercialized*

This project requires the use of Bluetooth technology for wireless transmission of data between the cane attachment, headset, and the Android phone. This transmission of data should protect user information and therefore must follow IEEE Standard 802.15.1™-2005 titled "13 Security".[17] This standard states: in order to provide usage protection and information confidentiality, the system provides security measures both at the application layer and the link layer. The primary concern is protection of user information, which Bluetooth technology addresses by incorporating authentication and encryption measures. Bluetooth technology incorporates four separate security measures during its operation on any device: a device address, two secret keys, and a pseudo-random number for each transaction. The address can only be obtained by the user of the device or by an inquiry from another device. The two secret keys are taken from the initialization process and are never revealed. The pseudo-random numbers are used for different security functions but are always randomly generated and therefore impossible to predict in advance. The devices must be manually paired initially by the user, ensuring protection of the device address. The application on the Android phone is coded to connect only to the two peripheral devices and nothing else, and if these are not paired it will not connect, thus maintaining specificity and user authentication.

This project requires the use of an SOIC microcontroller for controlling the peripheral devices. For the purposes of this project, the SOIC chip must conform to the standard outline MS-013 given by standard JEDEC Solid State Product Outlines.[18] MS-013 states the correct dimensions that a SOIC chip should follow as well as outlining various design elements. The SOIC used for this project meets the standard outlined by JEDEC, falling within the minimum and maximum range of specified dimensions.

This project requires the use of a Micro-B Universal Serial Bus for charging the batteries of the headset and cane attachment. The Micro-B Universal Serial Bus must conform to the specifications outlined in standard Universal Serial Bus 3.1 Specification, Revision 1.0.[19] Universal Serial Bus 3.1 Specification, Revision 1.0 states: a hub may provide power to all its downstream ports all of the time to support applications such as battery charging from a USB port. The Micro-B USB used for this project conforms to this standard by providing continuous power to support battery charging of the headset and cane attachments.

The project requires the use of microelectronic devices as components in the circuits built for the headset and can attachments. For this reason, the discussion of these microelectronic devices must conform to standard JESD99C Terms, Definitions and Letter Symbols for Microelectronic Devices.[20] JESD99C states: this standard will prove to be a useful guide for users, manufacturers, educators, technical writers, and others interested in the characterization, nomenclature, and classification of microelectronic devices. This project conforms to standard JESD99C by utilizing the correct nomenclature, symbols, and abbreviations outlined for use in schematics, equations, and descriptions.

4. Module Level – Requirements and Specifications

4.1. Headset & Cane Attachment Sub-Systems

Requirements for All Modules:

- Must run at 3.3V

4.1.1. Printed Circuit Board

Requirements:

- Must fit within the housing

Specifications:

- PCB Dimensions: 1.32" x 2.41"

4.1.2. Battery & Power Management

Requirements:

- Must supply steady voltage and current from varying battery voltage
- Must allow for the onboard battery to be recharged

Specifications:

- 3.7V 1000mAh Li-Ion battery
- Input voltage: 0.3V-5.5V
- Output voltage: 3.3V
- Max output current: 300mA
- Charge via micro-B USB connector

4.1.3. Ultrasonic Sensor

Requirements:

- Must be able to detect objects up to 7ft away with accuracy of 1in
- Must output data to the microcontroller
- Must deliver readings fast enough to stay within the desire response time
- Must have a well-defined, narrow detection range

Specifications:

- Operating voltage: 2.5-5.5V
- Range: 0-255in; Tolerance: 1in
- TTL serial output at 9600bps
- Reading rate: 20Hz
- Beam pattern[8] shown in ***Appendix I: Datasheets***

4.1.4. Accelerometer

Requirements:

- Must be able to determine the orientation of the device
- Must output data to the microcontroller

Specifications:

- Operating voltage: 2.2-3.6V
- Measuring range: $\pm$ 1.5g
- Measuring axes: 3 (x, y, z)
- Analog output between 0.8-2.4V[9]

4.1.5. Microcontroller

Requirements:

- Must receive data from the sensors and output data to the Bluetooth module
- Must send properly formatted data that can be interpreted in Android application

Specifications:

- Operating voltage (V_{CC}): 1.8-5.5V
- 2 full duplex USARTs, 12 10-bit ADC channels[10]
- Output formats: " HS %d\r HA %d\n" or " GS %d\n GA %d\n"
    - "HL\n" or "GL\n" when battery is below 20%

4.1.6. Bluetooth Transceiver

Requirements:

- Must receive data from the microcontroller, then transmit data to smartphone

Specifications:

- Operating Voltage: 3.3-5V
- Receives serial UART data and transmits via the Bluetooth antenna with configurable baud rate[11]

4.2. Android Application Sub-System

4.2.1. Bluetooth Connection

Requirements:

- The application should automatically connect only to the headset and cane attachment without prompting the user to manually connect
- Needs to receive serial data over the Bluetooth connection
- Should check if Bluetooth is enabled on user's android phone
- Should scan for Bluetooth enabled devices
- Should get list of paired Bluetooth devices
- Should check if "headset" and/or "cane" are in acquired list of paired devices
- Should check if headset and/or cane are connected after initiating a connection and after every audio alert

Specifications:

- Peripheral devices will use Bluetooth 2.1 wireless technology to communicate with the smartphone
- Connection will use the Bluetooth SPP (Serial Port Profile)
- Will use getPairedDevices() function to acquire list of paired Bluetooth devices
- Will use startDiscovery()function to scan for Bluetooth devices within 10 meters
- Will use isEnabled()function to check if Bluetooth is enabled on phone
- Will use getName()function to check if headset/cane is in list of paired devices
- Will use connect()function to connect to the headset/cane attachments
- Will use isConnected()function to check if headset/cane are still connected

4.2.2. Data Processing (Algorithm)

Requirements:

- Must distinguish between headset/cane data, ultrasonic and accelerometer
- Must determine if objects are in users path at head-level and ground-level
- Must determine if user's head is tilted up/down and/or if user's cane is properly oriented
- Must output formatted notifications to text-to-speech module

Specifications:

- Inputs taken from Bluetooth module will be parsed looking for input message formats: " HS %d \r HA %d\n", "HL\n"; " GS %d\r GA %d\n", "GL\n"
- Will determine from data if head-level object is within 0-5ft range or ground-level object is within 0-7ft range
- Will determine from accelerometer data if the headset is beyond 20˚ from level and/or if the cane attachment is rotated more than 30˚
- Output notification formats: "Hx.x", "Gx.x", "CW", "CCW", "HC", "GC", "HL", "GL"

4.2.3. Text-to-Speech Notification Generator

Requirements:

- Should alert the user to obstacles on the ground with distance to the object
- Should alert the user to head-level obstacles with distance to the object
- Should alert the user when the cane is improperly oriented
- Should alert the user once the path is clear after an obstacle was detected
- Should alert the user when battery in either of the peripheral devices is low

Specifications:

- Will alert user with "*Enable Bluetooth*" if Bluetooth is not enabled on phone
- Will alert user to ground-level objects with "*Ground object X feet ahead*" audio output when a ground level obstacle is detected by the ultrasonic sensor
- Will alert user to head-level obstacles with "*Head obstacle X feet ahead*" audio output when a head level obstacle is detected by the ultrasonic sensor
- Will alert user with "*Head* or *Ground Clear*" audio output if a head level or ground level obstacle is no longer in the user's path
- Will alert user with "*Rotate cane left* or *right*" when cane is improperly oriented
- Will alert user that "*Headset* or *Cane, low battery*" when battery is below 20%

5. Individual Module Design

5.1. Headset and Cane Attachment Sub-Systems

Both the headset and walking cane attachment are very similar in their operation. They will both be described and broken down into modules here simultaneously because each module applies to both devices. **Figure 15.18** shows the final schematic of the sub-system circuit.

5.1.1. Printed Circuit Board (PCB)

The circuitry for both peripheral devices must be soldered onto a single PCB for each device. The PCBs will be simple 2-layer PCBs integrating all of the necessary components for the devices to operate except the ultrasonic sensor and battery. The PCBs must fit within the housings so they have been designed to match the width of the battery at 1.32” x 2.14”. The boards must be laid out in such a way that allows easy access to both the micro-B USB connector and the on/off switch. The holes in the housing must then line up with the location of where those components are placed. ***Appendix F: Board Fabrication Details*** shows the finished PCB design and all components.

5.1.2. Battery & Power Management

The power management module of the devices includes the battery pack as well as the circuitry necessary to both charge the battery and regulate the output to 3.3V. The 3.7V 1000mAh battery was conservatively chosen for its ability to power the device well beyond the specified 24 hours. It can also be easily regulated to 3.3V. The PowerCell[15] from Sparkfun was chosen as the best commercial option that includes both the LiPo-battery charging and boost converter. The boost converter circuit uses a low input voltage synchronous boost converter IC, TPS61201DRC[12], capable of taking inputs as low as 0.3V and maintaining a steady 3.3V output. It also has an undervoltage-lockout (UVLO) included to ensure that the battery is never drained below 2.6V, to prevent battery damage. The battery output is connected to one of the ADC ports of the microcontroller through a 10MΩ current-limiting resistor while monitoring the battery voltage. When it goes below 2.82V, or 20% of the usable voltage range left, an alert must be sent out.

The charging of the battery is controlled by a dedicated lithium-polymer charge management controller, MCP73831[13]. The datasheet provides a typical application of the IC, shown below, that is used. The V_{in} comes directly from the micro-B USB connector.

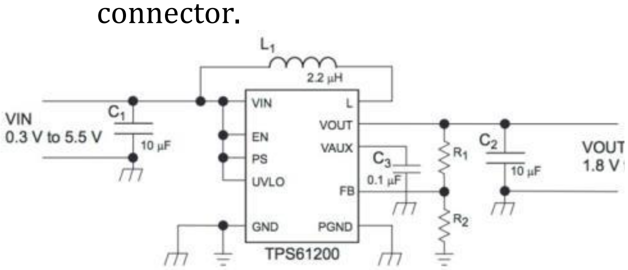


Figure 5.1: Sample 3.3V boost converter circuit.[12]

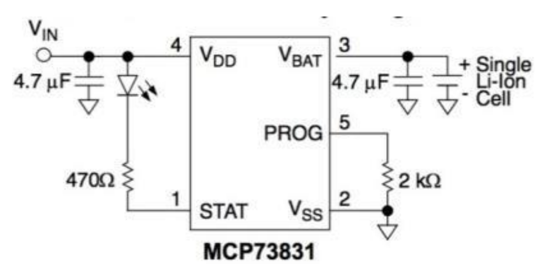


Figure 5.2: Typical charging circuit application.[13]

5.1.3. Ultrasonic Sensor

An ultrasonic rangefinder was chosen as the best solution for detecting obstacles for a variety of reasons. This sensor operates by sending out ultrasonic sound waves and calculating the distance to the nearest object by measuring the time it takes to receive the wave back. The ultrasonic sensor is capable of meeting the specified distances with a maximum range of 255in (21.25ft) and 1in accuracy. It also operates from 2.5-5.5V, so it will be able to run on the desired low voltage battery. Finally, ultrasonic sensors can commonly attain at least a 20Hz reading rate,[8] which equates to a new reading every 50ms. This will allow the system to keep its response time below the specified 100ms.

The ultrasonic sensor has three types of output, RS232, analog, and PWM. The digital RS232 serial output at 9600bps was used for its convenience when interfacing with the microcontroller. An inverter was placed between the output pin of the ultrasonic sensor and the input pin of the microcontroller to convert the RS232 signal to standard TTL values.

It is also important to note that an IR sensor was considered for the headset because it can sense objects up to 5ft away as well.[14] After some testing, it was determined that the ultrasonic sensor performed more accurately than the IR sensor. Long-range IR sensors also require higher voltages, at least 4.5-5.5V, making it a poor choice for a low-voltage embedded system.

5.1.4. Accelerometer

The accelerometer module will be responsible for determining the orientation of the user's head as well as the orientation of their walking cane. The accelerometers will detect changes in the static gravity acting on the x, y, and z-axes and output analog voltage signals. Only a single axis is used in this implementation because it supplies enough information to determine whether a user's head is tilted up or down as well as the rotation of the walking cane. More elaborate packages that include gyroscopes coupled with accelerometers were deemed unnecessary, as the accelerometer alone was found to provide enough information.

The accelerometer outputs a linear analog voltage between 0.8-2.4V for –g to g, which can be read by the microcontroller's ADC channels. The accelerometer was used as specified in its datasheet with a 3.3nF capacitor between the output and ground. There is also a 0.1μF capacitor between V_{DD} and ground. The sleep pin is permanently pulled high through a 10kΩ resistor, which allows the accelerometer to operate constantly. A current buffer was placed between the analog output of the accelerometer and the input ADC pin of the microcontroller because the accelerometer has a relatively high 32kΩ output impedance. The buffer ensures that the value read by the microcontroller is accurate.

5.1.5. Microcontroller

Both devices require a central hub capable of processing the data coming from the sensors and sending that data out to the Bluetooth transceiver. This particular system will require low voltage microcontrollers to minimize the current draw and keep battery life as long as possible. In addition, running the microcontroller at a relatively low clock speed such as 4MHz as opposed to 12MHz will greatly reduce the current draw of the microcontroller. **Figure 15.41** shows the supply current versus clock frequency at various voltages.

The microcontroller will be responsible for reading from the ultrasonic sensor and accelerometer, interpreting the data, and formatting it to be sent to the Bluetooth transceiver in such a way that can be easily understood in the Android application. The data will be formatted in messages such as " HS %d\r HA %d\n", or "HL\n". The 'H' would change to a 'G' for the cane attachment. This corresponds to headset sonar and ground sonar with a distance in inches and the accelerometer axes with degrees. This is to ensure that the Android application knows which sub-system the data corresponds to. The coding consists of three main parts: the main loop, a serial receive interrupt, and an ADC complete interrupt. The main loop constantly starts a new ADC conversion of the accelerometer output voltage, while the interrupts trigger when there is new data from the ultrasonic sensor and when the ADC completes. The full, commented, and annotated code can be found in ***Appendix H: Source Code*** as well as an execution flowchart showing the logic of the code in ***Appendix B: Software Flowchart***.

```
void initialize(void);
void start_adc(void);
unsigned char decode_number(unsigned char x);
ISR(USART0_RX_vect)
ISR(ADC_vect)
ISR(TIMER1_OVF_vect)
```

Figure 5.3: Function prototypes and interrupt vectors for microcontroller code.

5.1.6. Bluetooth Transceiver

It was decided that both peripheral devices would need to be wirelessly connected to the smartphone to make the system as easy as possible to use. A wired connection between the smartphone and a device mounted to the end of a user's walking cane would be extremely cumbersome. In addition, having two devices wired to a smartphone that generally has only a single port would be a difficult challenge to overcome. A wireless solution was the only practical way of integrating both devices with the smartphone. Bluetooth is the current predominant wireless technology in embedded systems and will be used as the means of communication between the peripheral devices and the smartphone.

5.1.7. Key Component Selection

The Power Cell LiPo Charger/Booster[15] was chosen as the best option for power management because it includes all of the necessary circuitry in a rather concise package. The main PCB for the system is designed so that the PowerCell PCB could be easily mounted on top of it.

The Maxbotix LV-MaxSonar-EZ3[8] ultrasonic rangefinder was chosen as the best possible ultrasonic sensor for this system. For one, the LV series of sensors from Maxbotix are low power devices, using only 2.0mA of average current as opposed to 3.4mA in their XL series. The LV-EZ3 also can also operate at up to 20Hz, whereas the XL series maxes out at 10Hz. This will help keep the response time of the system as small as possible. Within the LV series, the EZ3 sensor was chosen because its narrow beam pattern and good side object rejection best met the desired 20-degree horizontal detection range.

The MMA7361L[9] triple axis accelerometer was chosen for a variety of reasons. The output of this particular accelerometer is analog, which is generally accurate and easy to interface with the microcontroller via the multiple analog to digital conversion channels. The analog accelerometer also has an extremely low current consumption of up to 400μA.

The ATtiny1634[10] microcontroller has been chosen for both peripheral devices. First, the ATtiny series was looked at exclusively because they are much smaller than AVR's larger ATMEGA series. Keeping the devices as small as possible is crucial to the overall design. The ATtiny1634 was chosen, in particular, because it is the only ATtiny microcontroller with two full duplex USARTs. This will allow both the ultrasonic sensor and Bluetooth transceiver to be directly connected to independent USARTs.

Finally, an HC-05 Bluetooth transceiver module[11] was chosen to handle the Bluetooth communication. This specific Bluetooth transceiver is extremely low cost and offers a great deal of functionality to the end user. Any Bluetooth transceiver allowing serial RX/TX type data transmission would work in this instance.

5.2. Android Application Sub-System

5.2.1. Bluetooth Module

Bluetooth technology is used in this system for the purposes of transmitting data from the headset and cane attachment to the Android application. Data is only to be sent via Bluetooth to the application, not from it, and the application generates alerts depending on the data received. Sending data over Bluetooth allows the establishment of a connection not reliant upon an Internet source, making data transmission more efficient than alternative methods.

The Bluetooth connection is executed first when the application starts. The code starts with checking if Bluetooth is enabled on the user's phone using the isEnabled() function. This function returns "True" if Bluetooth is enabled on the user's phone, in which case it moves on the scan for Bluetooth devices within a 10 meter radius using the startDiscovery() function. If Bluetooth is not enabled on the user's phone, the application outputs an audio alert telling the user to enable Bluetooth and again uses the isEnabled() function to check if Bluetooth is enabled.

After scanning for nearby Bluetooth devices, the code moves on to getting a list of devices already paired with the phone using the function getPairedDevices(). From here the code checks if the headset is paired with the user's phone using the getName() function. If the "headset" is found in the list of paired devices then the program moves on to connect the headset with the user's phone using the connect() function. If the headset is not paired with the user's phone, the application outputs an audio alert telling the user that the headset is not paired and then the code loops back to the start of the program. After calling the function to connect the headset and user's phone, the code checks if the two are indeed connected by using the isConnected() function. If the two are in fact connected the code moves on to getting the list of paired devices again. If the two are not connected, the application outputs an audio alert telling the user that the headset is not connected and then the code loops back to the start of the program. After the application again gets a list of paired devices, it then uses the getName() function to check if "cane" is in the acquired list of paired devices. If the "cane" is not in the list of acquired paired devices, the application outputs an audio alert telling the user that the cane is not paired and the code then loops back to the start of the program. If the cane and the user's phone are paired, the code then calls the connect() function to start a connection between the two devices. When a connected socket is returned, the code then moves to check if the cane and user's phone are in fact connected using the isConnected() function. If they are not connected, the application outputs an audio alert telling the user that the cane is not connected and then the code loops back to the start of the program. The Bluetooth functionality available in this application is heavily built around the sample provided by Google under the Android Open Source Project, BluetoothChat example.[21]

5.2.2. Data Processing (Algorithm) Module

The algorithm controls the exclusively audio user interface. The algorithm deals with objects' distances, head and cane orientation, system design parameter checks, and alert creation. The algorithm will take data received via Bluetooth connectivity from peripheral devices as its input stream of data. Using this stream, the algorithm will separate the pieces of data into sets of data in relation to each peripheral device and the sensor to which they correspond. Each of the sensors on the peripheral devices provides different feedback that becomes useful to the user. Additionally, the algorithm handles conveying low battery information to the user if either battery is below 20%.

Initially, the data stream is received by the Bluetooth module of the Android device one character at a time (i.e. 'H' 'S' '0'), meaning, for example, that a set characters beginning with 'H''S' or 'G''S' and ending with '\n' is one unit of usable information regarding distance. The algorithm distinguishes between the head and ground-level from the first character (H or G). From this, the algorithm knows that the next characters pertain to information regarding head-level or ground-level attachments.

Separation into sensor types is done by looking at the second character, 'S' or 'A' or 'L', signifying that unit of information pertains to the sonar, accelerometer, or battery, respectively. For the ultrasonic sensor, the next one to three characters provides the numerical information in inches and the last character will be '\r' signifying the completion of one set of useable information. Similarly, for the accelerometer sensor, the first one to four characters after the 'A' provide the numerical information in degrees about peripheral orientation and the last character '\n' is the end of the set. Once a unit of data containing both ultrasonic sensor and accelerometer data is obtained, the unit is separated at its spaces into four segments of data and stored in an ArrayList. Within the ArrayList, the distance and angle can be found and converted from strings to numerical while also scaling the distance to foot values. The numerical information from the ultrasonic sensors and accelerometer sensors are run through a 10-point moving average as new information is received.

After the 10-point moving averaging is complete, the system parameter checks are initialized, starting with the accelerometer section. If the head angle is between ±20°, the algorithm will proceed to checking the headset distance value. Similarly, if the cane orientation is between ±30°, it will proceed to checking the ground distance value. For both, if outside the degree range, the distance value will not be checked so no obstacle alert can be triggered. However, there will be an output to rotate the cane either clockwise or counter-clockwise. If either degree value is within the desired bounds, the algorithm will check if the corresponding distance value is in the alert range for the headset (0-5ft) or cane attachment (0-7ft).

If these conditions are met, a distance message is created and passed to the text-to-speech module for each peripheral device. The algorithm will only re-alert the user once the obstacle is half a foot closer. Additionally, all alerts will be given in a sequential order as they are received. At the initialization of the algorithm, "clear" variables called CLEARH and CLEARG are set to true. Once an alert at either peripheral device is created, the respective "clear" variable is set to false. In the next iteration, if the object is out of the device range, then the respective "clear" variable is set to true and a CLEAR message is created.

In terms of the battery, once the algorithm reads 'L' after the first character it creates a battery low power message for the specific peripheral device, while also looking for the 'n\' character to know that the unit of data is finished.

A completely detailed execution flowchart for this algorithm can be found in ***Appendix B: Software Flowcharts*.**

5.2.3. Text-to-Speech Notification Generator

This module utilizes built-in functionality[16] of the Android operating system and Android SDK to convert text notifications into spoken audio alerts for the user of the system. The module needs to receive the desired notification from the data processing module, convert it to speech, and output the alert to the user. Since the user of this application is visually impaired, the only practical way of notifying them is an audio alert. A simple vibration would have sufficed had our application only needed one notification but for the amount of information we provide, audio alerts are necessary.

The text-to-speech generator represents the third major module of the Android application after the Bluetooth connection module and the data processing module. The text-to-speech generator receives a formatted message from the data processing module and then through a series of decisions generates an audio alert to notify the user. There will only be a distance alert when there is at least half a foot of change reported by the sensors at either head or ground level. The text-to-speech generator is also set up so that a new message will not be spoken while it is already speaking something. Therefore, it will never be interrupted mid-alert. Another important change that was made to the text-to-speech built into the Android SDK was to speed up what was being said as well as reducing the overall number of words to be spoken. These messages happen in real time and are constantly generated by this module whenever an alert is necessary.

Sample alerts include collision alerts at the head and ground level to assist the user in avoiding those obstacles. Another alert will be is necessary to have the user turn the cane left or right based on the current orientation of the cane attachment. Based on the values returned by the accelerometer, a decision can be made as to the orientation of the cane attachment and how it must be rotated to ensure it is properly oriented. Also, an alert is included to notify the use when both the head area and ground are clear of obstacles. This alert is spoken after an obstacle was already detected to notify the user that they have found an open area to continue moving forward. Finally, a low battery alert is included for both the headset and cane attachment when the battery's voltage drops below 20%.

A completely detailed execution flowchart for this module can be found in ***Appendix B: Software Flowcharts***.

6. Module and System Tests

6.1. Overall System

In terms of the overall system, the Collision Avoidance System must meet multiple criteria before the team can be fully satisfied with the product. These tests for the overall system will correlate to the detection of obstacles at various distances, the effectiveness of ignoring non-relevant objects (i.e. ceiling), response time for various gaits, working angle boundaries, and finally the 24-hour use time of the system.

To test detection of obstacles at various distances, the system will be placed over 10ft away from a wall, then incrementally moved closer to verify that the object will only be detected when it is within the specified 7ft range at ground-level and 5ft range at head-level. This will prove the system's viability for large, solid objects. In similar tests, either a broom handle (emulating a small branch) will be placed at head level, 10ft in front of the user, or a 1ft tall box will be placed at ground-level 10ft in front of the user. Distances will be marked off to ensure the system outputs the proper distance to within 0.5ft.

Another necessary test is determining the angle boundaries of the sensors and the system's ability to ignore objects above the user's head. For this test, the headset will be placed next to a protractor to determine the angle at which notifications cut off. The device will be slowly tilted back and any solid object will be moved in and out of its field of view. The test will be successful if there are no notifications after being tilted 20° in either direction. A similar test can be performed for the cane attachment, only this time rotating it past 30° in either direction. A test will also be performed where a solid object is moved in from the sides of the user to determine the angle at which it is first detected as an obstacle. For success, this value should be 10°.

Response time will be tested by dropping a solid object in front of either sensor and starting a timer at the same time. No more than 100ms should pass between the object entering in front of the system and the start of the audio alert. Another way of testing this would be to insert timestamps for testing purposes on specific alerts to compare the time between when an obstacle is detected and when the notification is triggered.

Finally, testing the 24-hour power range will be done by leaving the system on for 24 hours. Then, if the peripheral devices are still running, the test would be continued to see how long the battery will last during continuous use.

6.2. Key Modules

Level 1 Testing:

In terms of the level 1 testing of the complete sub-systems that comprise the Collision Avoidance System, the main goal would be the integration of each module within the sub-system. The peripheral devices will be tested individually prior to integration with the Android application. For the sub-system tests, the headset and cane attachment must show that they can output distance, degree, and battery alerts to the Android application. Because a connection has already been shown to work between the devices and a laptop computer, they can be connected again to show that the devices are outputting data in the format specified above. If each device outputs distance and degree information as well as battery alerts when the battery is low, the test will be successful.

In terms of the Android phone, at this level the phone must be able to receive data from two different Bluetooth connections simultaneously. This functionality can be tested as it was in the prototype demonstration by connecting to two additional phones. To test this, not only would two different connections need to be established, but the phone would also need to show two separate sets of raw data coming from the two sources.

Level 2 Testing:

With relation to level 2 testing, components that will be examined are the individual modules that make up each sub-system: the sensors, power management, Bluetooth transceiver, Bluetooth connection, data processing algorithm, and the text-to-speech-notification generator.

Testing the sensors will consist of multiple steps. The devices will need to be reprogrammed with specific code to test the modules and create a connection capable of outputting results. In terms of the ultrasonic sensor, verifying that the distances reported by the sensor match the actual values will be accomplished by measuring a solid object at various distances between 0-7ft using a measuring tape as reference. Similarly, testing the accelerometer can be achieved by using a protractor as reference, which will allow the displayed measurements of degrees from the zero point of each axis to be compared with real world values. These tests were actually already performed and the results can be found in

The power management circuitry will be tested by simply connecting a low input voltage of 1V and measuring the output voltage to ensure that the boost converter is working and outputting 3.3V. The charging circuit will be tested by attaching a slightly drained Li-Ion battery, which is initially at 3V. The battery will be left to charge for an hour, then tested again to see that they voltage has risen.

Additionally, tests must be run to verify that data from the microcontroller can be transmitted wirelessly over Bluetooth. To test such a Bluetooth module, predefined data such as a simple message "Hello" will be transmitted to a laptop and read to ensure what is sent is the same as what is received.

Testing the algorithm will be done using "fake" data sets simulating various scenarios such as normal walking towards an object, a car passing through the user's path (testing a quick object motion), a person passing through the user's path (a slow object motion), objects moving in/out of user's path, etc. These "fake" data sets will be created by using a conservative estimate for walking speeds of 3ft/s to 5ft/s. The premise of all the "fake" data sets is to change distance values from outside of the working ranges to within the working ranges abruptly. The data processing algorithm will realize that objects become relevant, and then determine after the moving average if the objects have a chance of colliding with the user. Not all objects that become relevant will necessarily collide with the user. For example, if a car quickly passes across the user's path, the ultrasonic sensors would pick up information about the object in the millisecond range. Readings that last for a few milliseconds would be smoothed out by the moving average resulting in the algorithm not notifying the user. This situation can be accurate because the car would be out of the user's path before the user is in a range for collision. On the other hand, if a person is passing through the user's path or walks in the user's path, the ultrasonic sensors pick up information about the person for a longer time period (in the seconds range). As a result, the algorithm's moving average will provide numerous values concerning the person that the parameter checks can use to create alerts. The other scenarios will work in the same similar way, where if reading concerning an object last a significant time interval (in the seconds range) alerts will be created, but if the readings last a short interval and/or if the readings are not changing the moving average will not allow for alerts to be created.

To test that the algorithm will simulate a streaming of data character by character, the various scenarios will be created as data points concatenated together as strings within files. The algorithm will then go into each specific scenario file and read through the string character by character separating them by peripheral device and sensor types, and then convert strings to numbers where necessary. When one unit of useable data is received, the algorithm will do the aforementioned processes of averaging, parameter checking, and alert creating. The outputs of the algorithm will be alert messages in the format of "Hx.x", "Gx.x", "CW", "CCW", "HC", "GC", "HL", or "GL" concerning various situations of objects within ranges, object passing out of ranges, incorrect cane rotations, and battery power.

The first step to test both the Bluetooth module and the text-to-speech generator is to obtain two other devices besides the main Android phone that work via Bluetooth serial port profile. The application can then be run on the main phone after which it will tell the user that Bluetooth is not enabled on the phone. After enabling Bluetooth on the user's phone, the application can be run again in which case an audio alert should be generated telling the user that the phone is not paired with "headset" or "cane." Once this occurs, Bluetooth on the two peripheral devices need to be manually turned on and the name of one device should be changed to “headset” while the other should be changed to “cane.” After this, the program should be run again and the code should recognize the two peripheral devices and try to connect to each separately. If in this process a connection has failed, the user will be notified via an audio alert. If the connection is successful, the code should jump to receive data from the two peripheral devices and translate this data into an audio alert. From the device named “head,” if a text is sent over Bluetooth to the main phone reading “H4”, the main phone should generate an audio alert in the form of “Head level obstacle 4 feet away”. From the device named ground, if a text is sent over Bluetooth to the main phone reading “G5”, the main phone should generate an audio alert in the form of “Ground level alert 5 feet away”. This series of tasks tests all functionalities of two modules in the Android application and will prove that they are functioning correctly. The Bluetooth connection test sees if a Bluetooth connection can be initiated and maintained between the user’s phone and two peripheral devices without any manual connection from the user. The text-to-speech test sees if the main phone can generate specific audio alerts from data sent over Bluetooth of two peripheral devices.

7. Timeline Estimation and Milestones

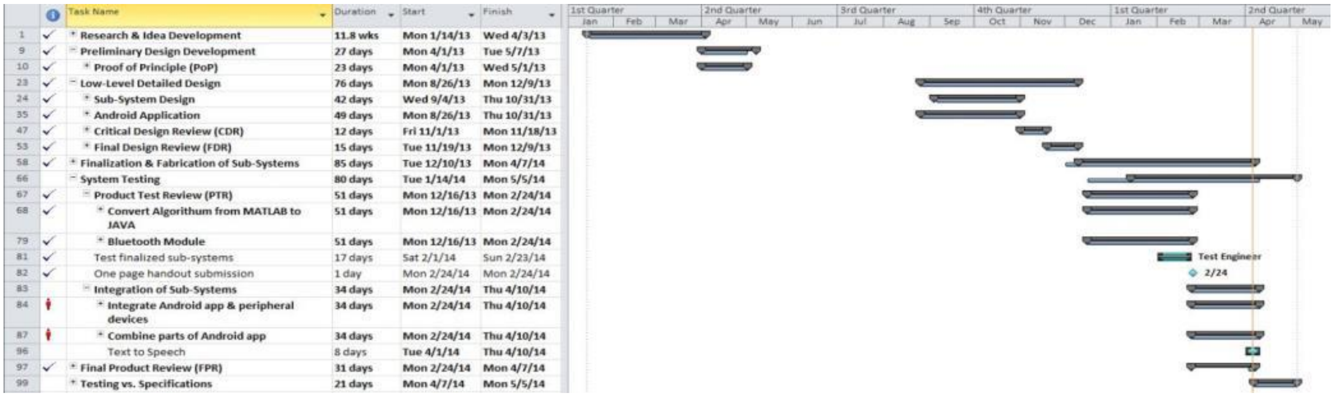


Figure 7.1: Gantt Chart.

The Gantt chart includes all major assignments, design decisions, and individual project tasks from the inception of the project idea to the projected completion of the device. The timeline starts in January 2013 and ends May 2014. The majority of the milestones include major deliverables (PDR, PoP, Prototype Demos, CDR, FDR, etc.). The full Gantt chart with all tasks and details can be found in ***Appendix C: Gantt Chart***.

8. Labor Costs Graph

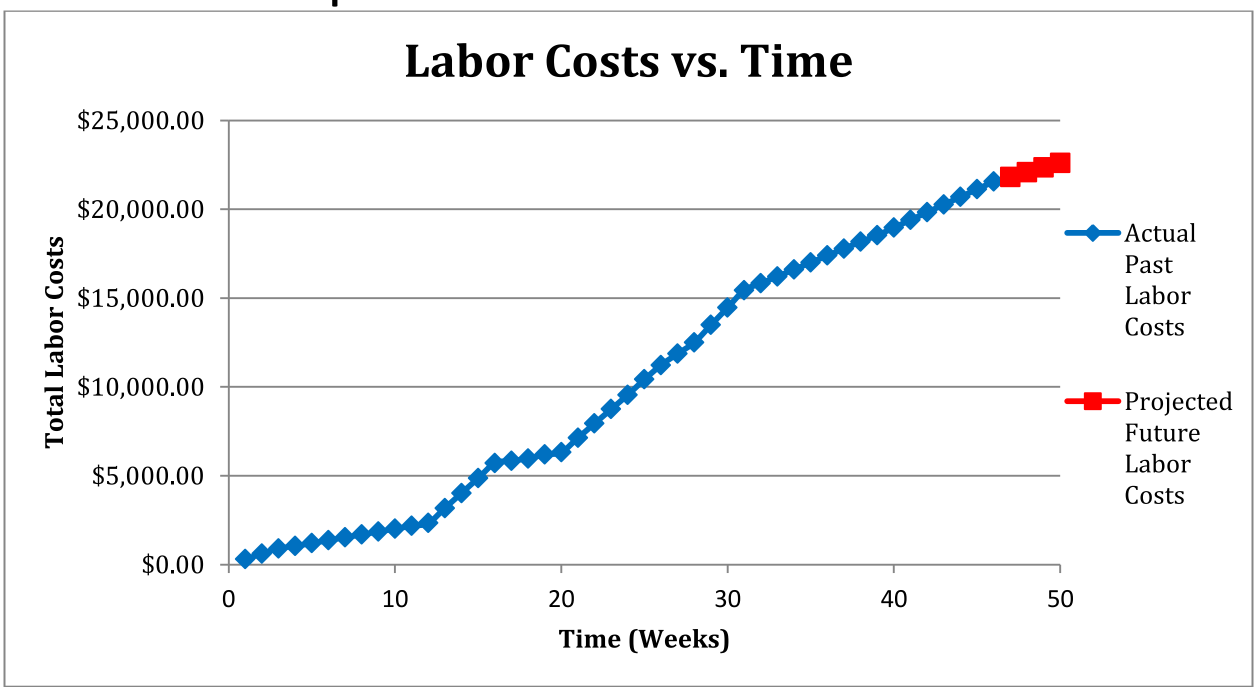


Figure 8.1: Labor Costs vs. Time Graph.

The labor costs vs. time graph includes an estimate of the labor required starting at the beginning of ECE3915 and estimates all the way until the end of ECE4925. The time span includes actual work weeks and excludes any vacation or interruption in the process of completing the project. The raw data used to estimate the overall labor costs is included in ***Appendix D: Economic Analysis Data***.

9. Economic Analysis

Part	Individual Cost	Total
(2) Ultrasonic Rangefinder – Maxbotix LV-EZ3	$27.95	$55.90
(2) Triple Axis Accelerometer Breakout Board – MMA7361	$11.95	$23.90
(2) HC-05 Bluetooth Transceiver Breakout Board	$9.28	$18.56
(2) ATtiny1634 Microcontroller	$1.72	$3.44
(2) 4MHz Crystal Oscillators	$0.46	$0.92
(2) SOIC to DIP Adapter 20-Pin	$3.95	$7.90
(2) 40-Pin Breakaway Headers	$1.50	$3.00
40-Pin Right Angle Headers	$1.95	$1.95
(1) AVRISP mkII	$36.59	$36.59
Total Cost		**$152.16**

Table 9.1: Cost of prototype parts.

The cost of each part for the prototype is outlined in the table above. The total for all parts of the prototype is $152.16. Adding in a 5% pass-through, the total for all parts of the prototype comes to $159.77.

Role	Hours	Salary
Project Manager	100	$66/hr
Design Engineer	32	$57/hr
Hardware Engineer	99	$48/hr
Software Engineer	141	$40/hr
Test Engineer	28	$36/hr
Technical Writer	93	$30/hr
Total	493	**$22,614**

Table 9.2: Labor hours and salary for each role needed for prototype.

The hours each member of the project team must contribute towards the prototype and the salary they will receive for doing so is listed in the table above. The hours for each role were derived from the labor costs vs. time graph by totaling the labor cost per role per week and dividing by that specific role's salary. The total hours needed to complete the prototype are 493 and the total labor costs are $22,614. After multiplying our total labor costs by a factor of 2.8 to convert our total salary costs to contract charges, the total labor cost is $63,319.20. Adding in the cost of prototype parts our total cost to build the prototype is $63,478.97.

In order for our prototype to be produced and sold, costs for the production of the device per unit must be calculated. These calculations explained in detail below.

Part	**Individual Cost**	**Total**
(2) ATtiny1634 8-bit microcontroller	$1.00	$2.00
(2) 3.7V LiPo-Ion battery - 1000mAh	$11.95	$23.90
(2) Maxbotix LV-MaxSonar-EZ3	$22.36	$44.72
(2) Three-axis low-g accelerometer (MMA7361L)	$6.36	$12.72
(2) HC-05 Bluetooth transceiver	$9.28	$18.56
(2) PowerCell Li-Po charger/booster	$15.96	$31.92
(2) Surface mount right-angle switch	$0.76	$1.52
(2) Single Schmitt-Trigger inverter	$0.103	$0.206
(2) 4MHz crystal oscillator	$0.21	$0.42
(2) Low voltage operational amplifier	$0.559	$1.118
(4) 20pF capacitor	$0.16	$0.64
(2) 3.3nF capacitor	$0.012	$0.024
(2) 0.1uF capacitor	$0.004	$0.008
(2) 10kΩ resistor	$0.09	$0.18
(2) 10MΩ resistor	$0.07	$0.14
(2) Printed Circuit Board	$7.95	$15.90
(2) Mini ball bearing	$1.52	$3.04
(12) PCB mounting screws	$0.10	$1.20
Total Cost		**$158.22**

Table 9.3: Estimated cost of production of device per unit.

The cost to price per unit of the produced device will be $158.22. The number of units likely to be sold is estimated to be 1000 units. For these 1000 units, the cost of production not including labor would be $158,220.00. The packaging cost for this device is estimated at $5.00 per unit sold so 1000 units would result in a total packaging cost of $5000.00. The total non-labor production cost for 1000 units is $163,220.00.

It is estimated that manufacturing verification will take approximately 200 hours, at a rate of $20/hr, which will result in a total of $4000.00 for the manufacturing process development and verification. For software verification, it is estimated that 350 hours will be needed, at a rate of $15/hr, which will result in a total of $5250.00 for software testing. The manufacturing verification and the software verification added together equal the production cost for labor, which is a total of $9250.00. Adding in a multiplier of 2.0 for indirect costs $9250.00 becomes $18,500.00.

The total cost for production includes the non-labor costs and the labor costs, which total to $181,720.00. After adding in overhead costs of 40% and a profit fee of 20%, the total cost for production becomes $181,720x1.4x1.2=$305,289.60. This costs expressed in term of per unit becomes $305,289.60/1000=$305.29.

The estimation for the total cost of the project is the cost of the prototype ($63,478.97) and the cost of production for 1000 units ($305,289.60) added together which equals $368,768.57. The total cost of the project in dollars per unit is $368,768.57/1000, which equals $368.7/unit.

To estimate the cost of distribution, first 20% must be added to calculate the wholesale price per unit: $368.77/unit x 1.2=$442.52/unit. Next, 50% must be added to calculate the retail price per unit: $442.52x1.5=$663.78/unit. The price is the final estimated retail price of our device is $663.78/unit.

10. Summary and Conclusions

Initially, the scope of the project was focused toward creating a facial recognition system for a security system of a house. A shift in direction for the project was implemented once it was realized that creation of a facial recognition algorithm was not an achievable goal in the timeframe and that use of a preexisting algorithm would result in the project being too simplistic. The scope of the project was directed toward creating an assistive device for the blind that would provide the user with information regarding facial recognition, currency, GPS, and object alerts. Again, through various design and achievable goal discussions, the project was slimmed down to be a collision avoidance system for the visually impaired focusing on using two peripheral devices and a phone to provide a user with information about objects in their walking path.

The data processing algorithm has been tested to ensure that it is capable of handling the necessary data and generating the specified results. Some progress has been made in getting simultaneous Bluetooth connections between an Android phone and peripheral devices; however, this has been the most difficult task so far in the project. The PCBs have been printed, populated, and tested to ensure that they are functioning as expected. The readings from the sensors match the preliminary data that was found in the prototype stage. Major progress has been made on the housings for both devices. Preliminary devices were 3D printed and have proven to be durable and lightweight, as expected. Refined versions were printed on a higher quality 3D printer and will be used in the final iteration of the system. The gimbal design for the cane attachment has also advanced with the use of mini ball bearings to ensure the sensor swings freely.

To this point, the group has been successful in implementing hardware prototypes that will eventually become the final device. Additionally, the sub-systems to be implemented for the final product were tested and finalized. The Bluetooth module has been the most challenging portion of the project to come together, and to this point, only one Bluetooth device can be connected to the device at a time while streaming data. With one device, the connecting portion of the integration process has begun, starting with creating a usable unit of data from a peripheral device and moving on to processing the portion of the data. Challenges faced through the preliminary integration stages include parsing the stream to a usable unit, clearing and overwriting the unit, speed of the acquisition of the data stream, and repetitive alert messages.

With patience and persistence many of the problems including parsing to usable units, clearing and overwriting the unit, and speed of acquisition have been solved and the repetitive messages are on their way to being solved. Moving forward, more time and effort will be spent on fixing the repetitive messages, but more importantly, the team must focus on establishing simultaneous connections to two peripheral devices via Bluetooth.

11. Qualifications of Key Personnel

11.1. Brandon Bernier

Brandon Bernier is a computer engineering major that has extensive coding knowledge and a great deal of experience with the C programming language as well as the design of both analog and digital circuitry. Brandon has taken multiple programming classes in C, in particular AVR based C coding, including Intro to C, Data Structures, and Operating Systems. More importantly, he has taken Intro to Microprocessors and Embedded Systems, which are based around Atmel's 8-bit AVR microcontrollers. In addition to knowledge of microcontrollers, Brandon is also very familiar with circuit design from Circuit Theory; Engineering Electronics; Digital Electronics and Design; and Design of Logic Systems I and II. These classes have given him a strong background in both high and low-level system design and a technical perspective with which he can approach this project.

11.2. Srinivas Tapa

Srinivas Tapa is a biomedical engineering major that has significant coding experience in C and MATLAB as well as being well qualified to design, simulate, and implement both analog and digital circuitry. Srinivas has taken multiple courses that in C such as Intro to C Programming and Data Structures. Additionally, Srinivas has a working knowledge of analog and digital circuitry from courses taken such as Circuit Theory; Engineering Electronics; Circuits, Signals, and Systems; and Design of Logic Systems I. Srinivas has also taken a course entitled Rehabilitation Medicine engineering, which studies various assistive categories and explores how devices can be made to meet the wants of the user, necessities of disability, and the regulations of the government (FDA). Courses taken throughout his college career have provided a range of skill sets that can be drawn upon to successfully implement the project.

12. Intellectual Contributions

The team will be responsible for writing all of the code necessary to ensure the completion of this project. To achieve this, the team will develop custom programs with the ability to convert the data provided by the sensors to useable information that can be relayed to the user in the form of audio alerts.

Brandon Bernier is in charge of the overall embedded system design, sensors, microcontrollers, battery and power management, and the design of the printed circuit boards (PCBs) necessary for the peripheral devices. Brandon has purchased ultrasonic sensors, accelerometers, and Atmel's ATtiny1634 microcontrollers. He is responsible for designing the circuitry and coding the microcontrollers. In terms of the PCBs, Brandon will initially develop the layout for the necessary boards in a computer program, then send the design out to be printed by a manufacturer. He will also assist in designing the 3D model for the 3D-printed housings. In addition, he will handle the Bluetooth connections in the Android application and ensuring it will work with two devices simultaneously. The majority of the Bluetooth code will be based upon the Bluetooth Chat example provided by Google on its Android website.[21]

Srinivas Tapa is in charge of algorithms and data processing for the Android application and the housing for the peripheral devices. Srinivas will develop the code and algorithm to differentiate between relevant and irrelevant objects as well as notifying the user to a clear path. Also, how often to notify the user about an object in their path and when batteries are low needs to be handled within the application. Finally, Srinivas will be in charge of all mechanical drawings of the housing and ensuring they get 3D printed. This also includes other components necessary to the housing design such as the gimbal and Velcro strap of the cane attachment.

13. Teaming Arrangements

The Collision Avoidance System is broken into the following components:

Brandon Bernier is responsible for the embedded system design and implementation, microcontrollers, sensors, battery and power management, PCBs, and testing. On the application side, he is responsible for creating the simultaneous Bluetooth connections so that the data provided from the peripheral devices can be properly processed.

Srinivas Tapa is responsible for creation of algorithms and data processing for the application, and housing/strap creation for the peripheral devices, including all mechanical drawings.

14. References

[1] http://www.who.int/mediacentre/factsheets/fs282/en/

[2] http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04539488

[3] http://img1.wfrcdn.com/lf/49/hash/4370/1757228/1/Deluxe+Folding+Blind+Cane.jpg

[4] http://www.disabilitystatistics.org/reports/census.cfm?statistic=1

[5] http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04539488

[6] http://www.ambutech.com/iglasses

[7] http://www.ncbi.nlm.nih.gov/pubmed/21820535

[8] http://maxbotix.com/documents/MB1030_Datasheet.pdf

[9] https://www.sparkfun.com/datasheets/Components/General/MMA7361L.pdf

[10] http://www.atmel.com/Images/Atmel-8303-8-bit-AVR-Microcontroller-tinyAVR-ATtiny1634_Datasheet.pdf

[11] ftp://imall.iteadstudio.com/IM120417010_BT_Shield_v2.2/DS_BluetoothHC05.pdf

[12] https://www.sparkfun.com/datasheets/Prototyping/tps61200.pdf

[13] https://www.sparkfun.com/datasheets/Batteries/UnionBattery-1000mAh.pdf

[14] https://www.sparkfun.com/datasheets/Sensors/Infrared/gp2y0a02yk_e.pdf

[15] https://www.sparkfun.com/products/11231

[16] http://developer.android.com/guide/topics/connectivity/bluetooth.html

[17] http://standards.ieee.org/findstds/standard/802.15.1-2005.html

[18] http://www.jedec.org/sites/default/files/docs/MS-013E.pdf

[19] http://www.usb.org/developers/docs/

[20] http://www.jedec.org/sites/default/files/docs/JESD99C.pdf

[21] https://android.googlesource.com/platform/development/+/25b6aed7b2e01ce7bdc0dfa1a79eaf009ad178fe/samples/BluetoothChat/src/com/example/android/BluetoothChat/BluetoothChatService.java

15. Appendices

Appendix A: Mechanical Drawings

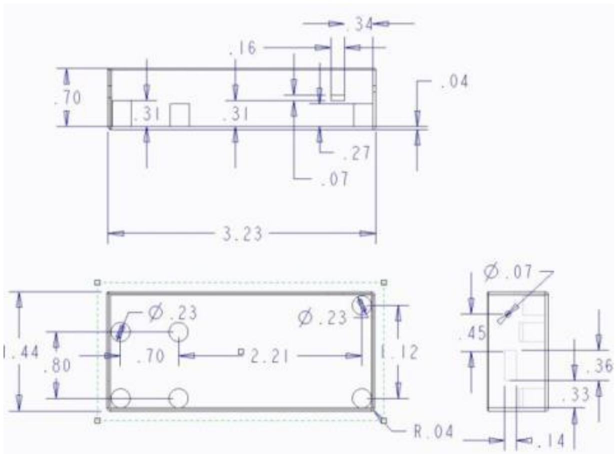


Figure 15.1: Headset mechanical drawing.

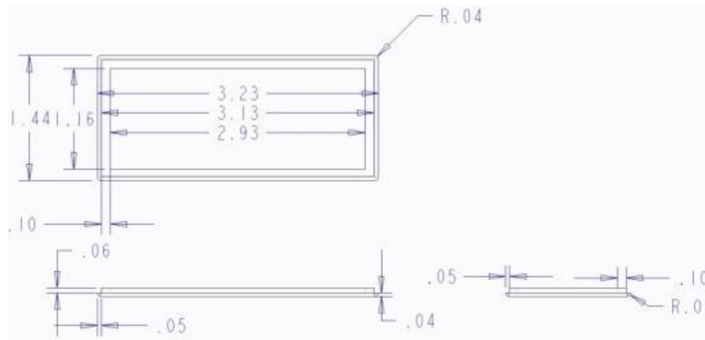


Figure 15.2: Headset lid mechanical drawing.

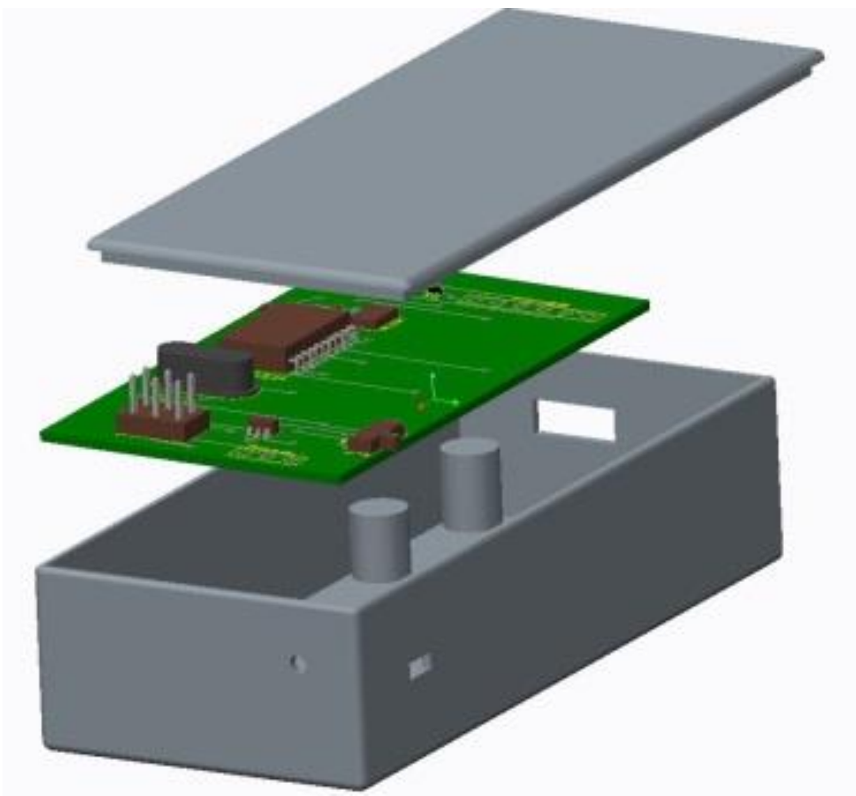


Figure 15.3: Headset attachment 3D rendering.

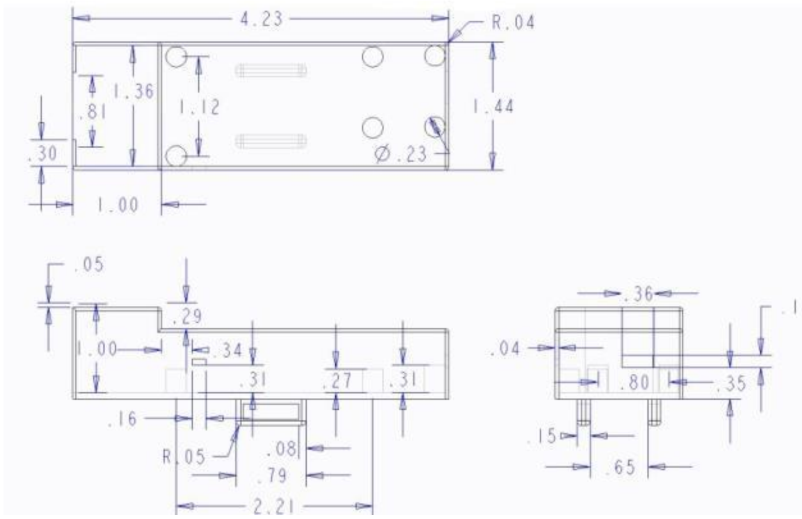


Figure 15.4: Cane attachment mechanical drawing.

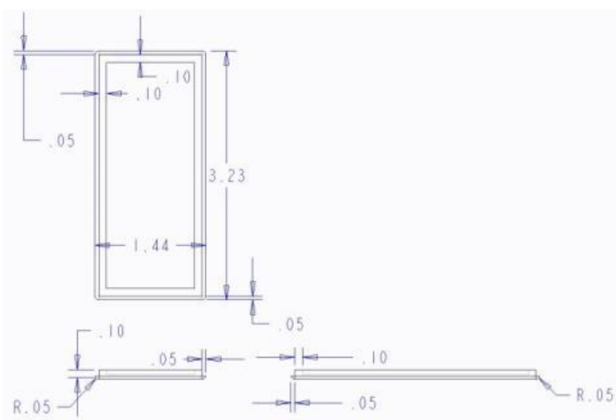


Figure 15.5: Cane attachment lid mechanical drawing.

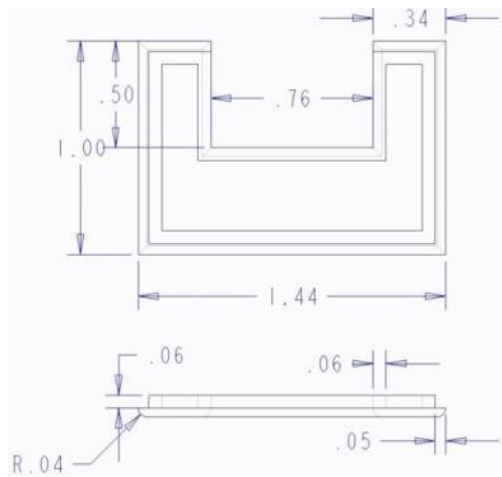


Figure 15.6: Cane attachment lid mechanical drawing.

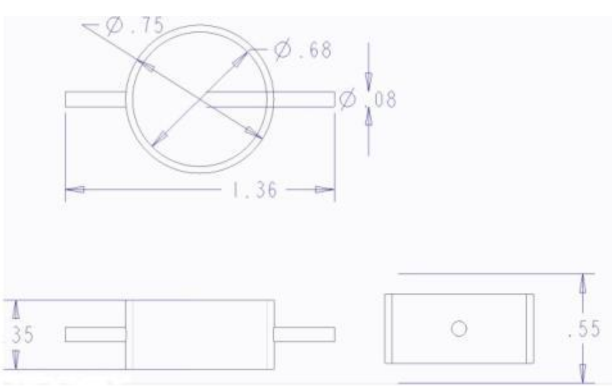


Figure 15.7: Sensor casing mechanical drawing.

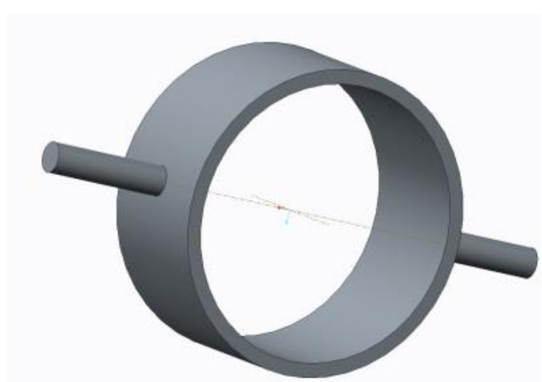


Figure 15.8: Sensor casing 3D rendering.

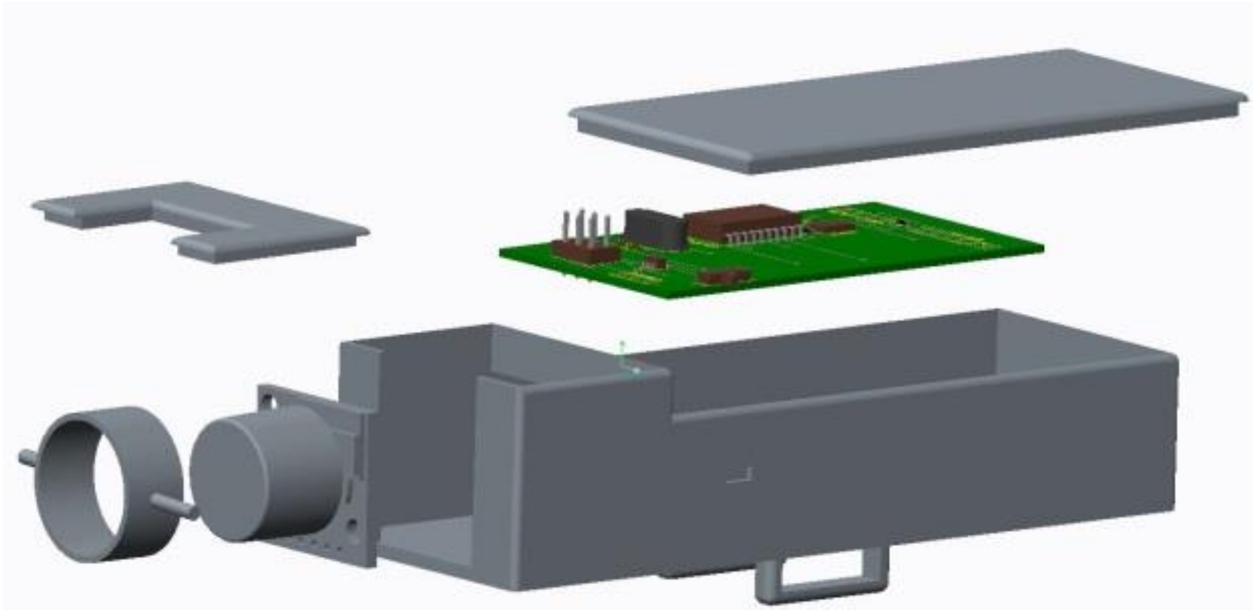


Figure 15.9: Cane attachment 3D rendering.

Appendix B: Software Flowcharts

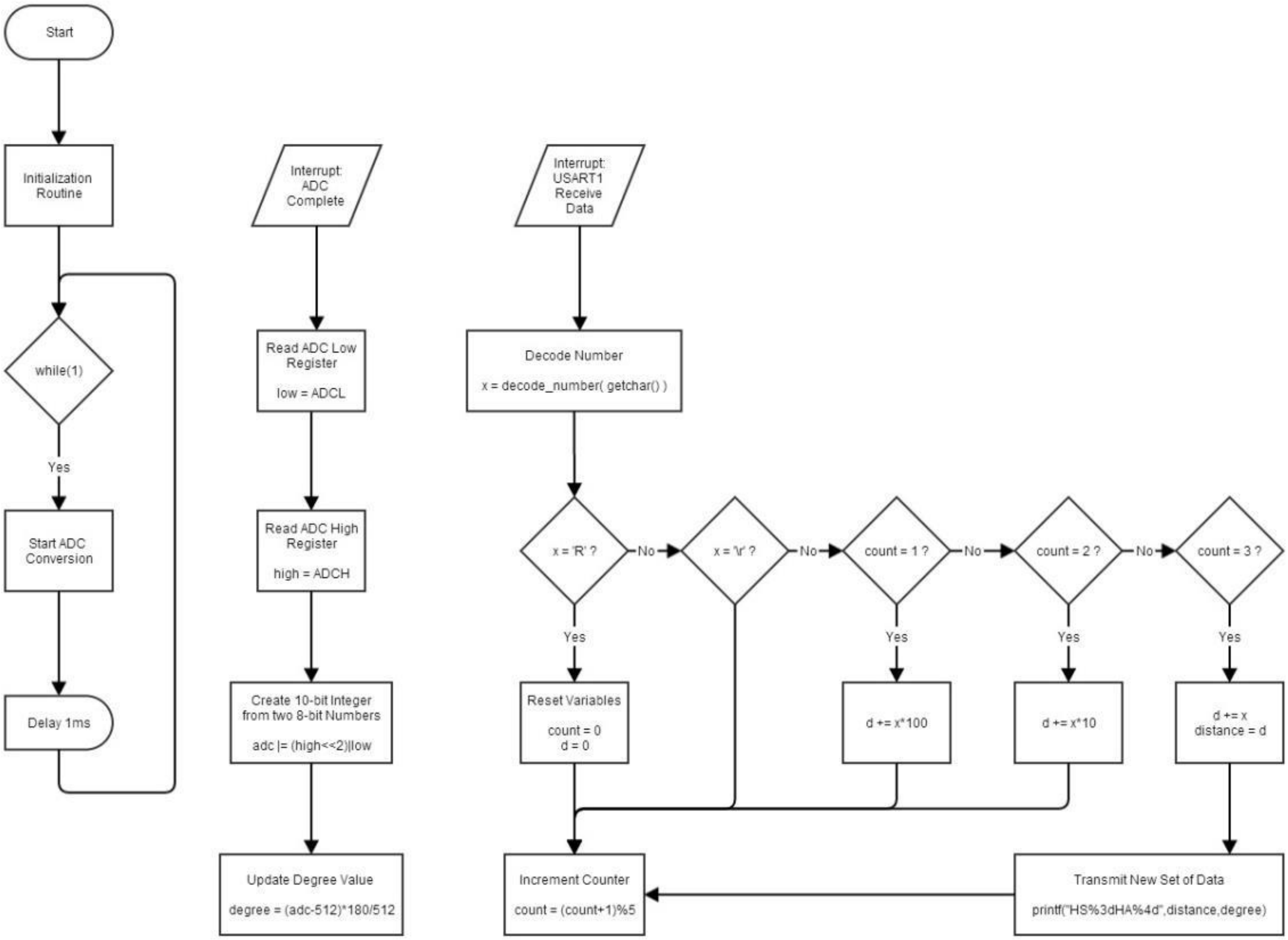


Figure 15.10: Headset/cane attachment execution flowchart.

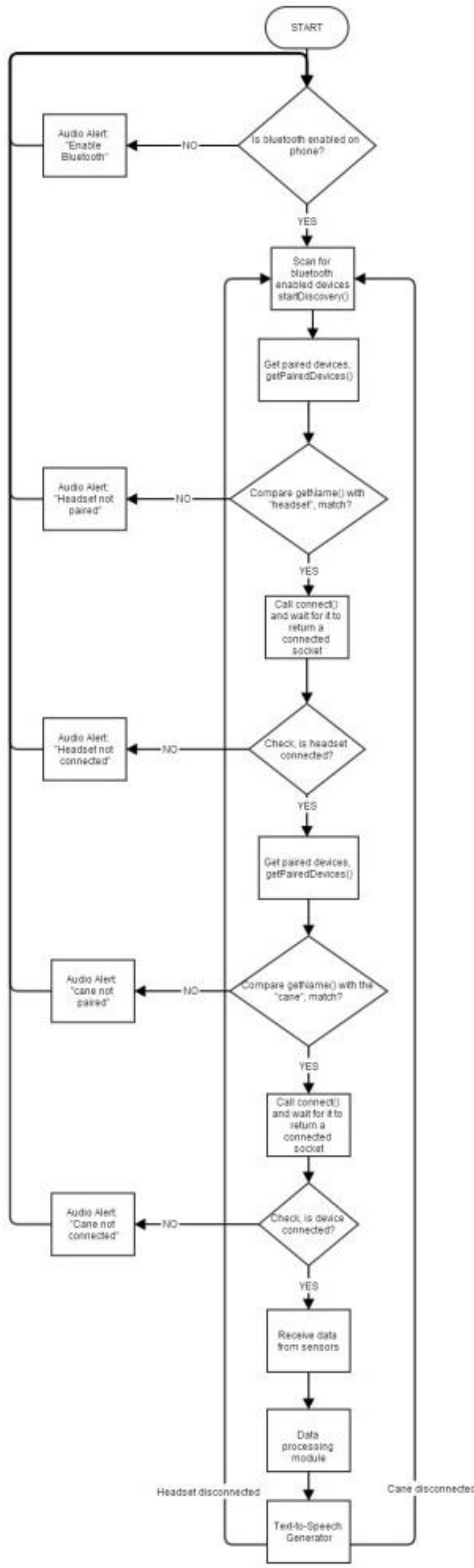


Figure 15.11: Execution flowchart for Bluetooth module.

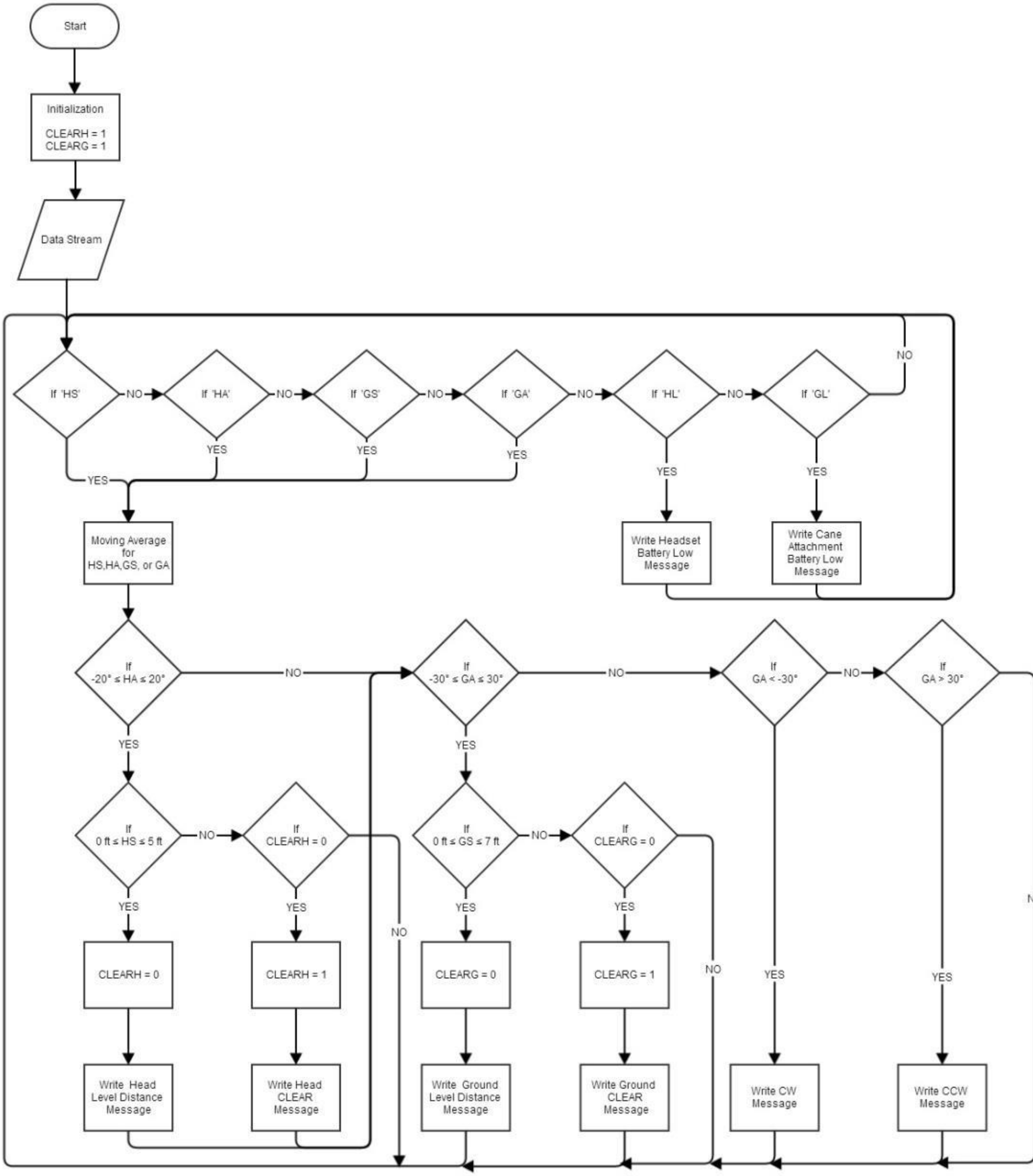


Figure 15.12: Data processing algorithm execution flowchart.

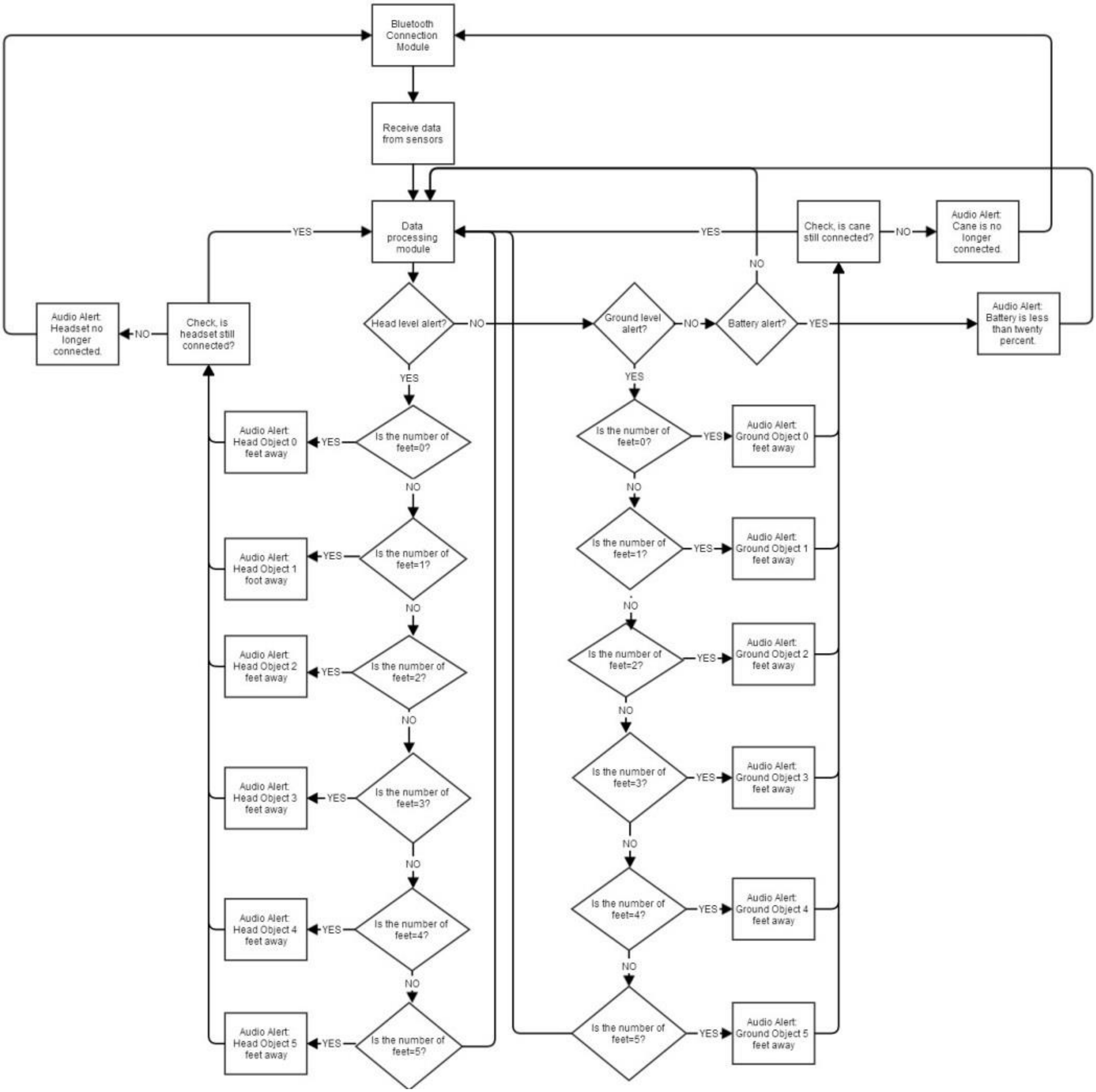
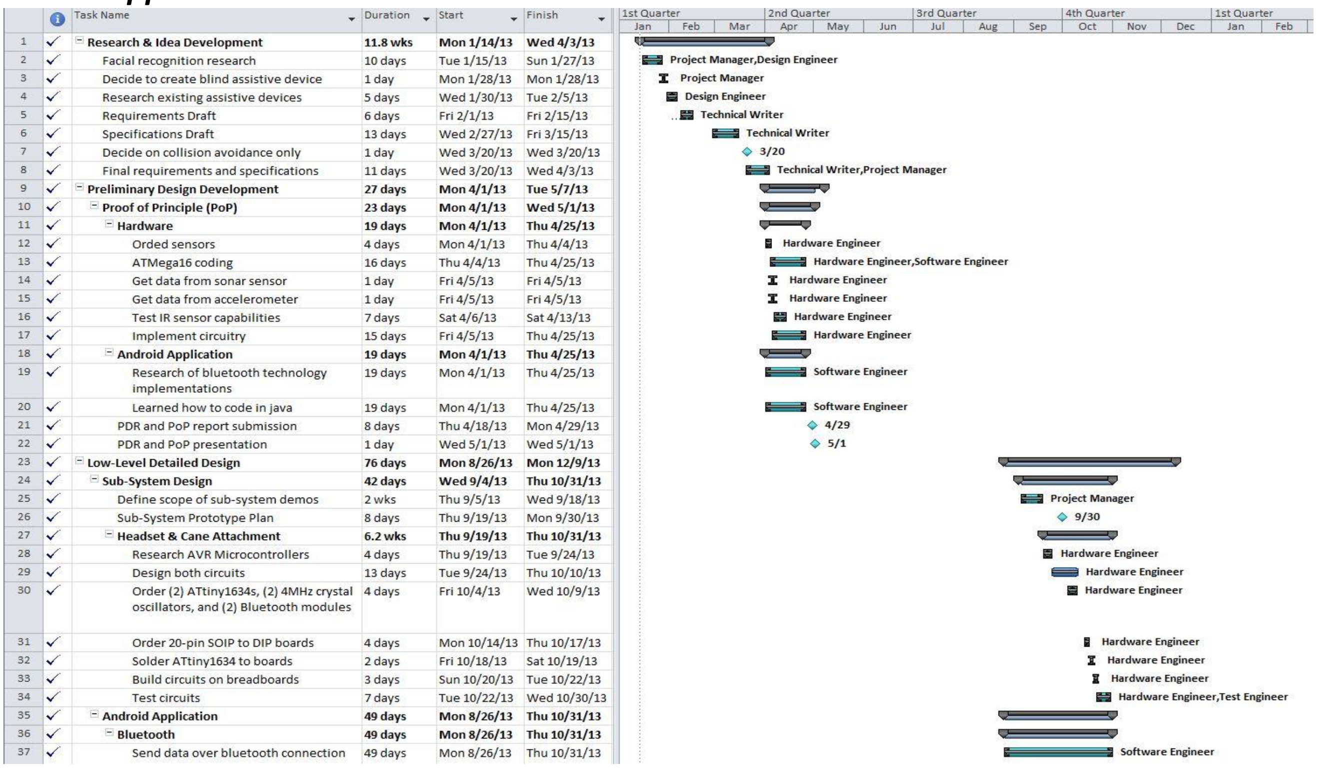


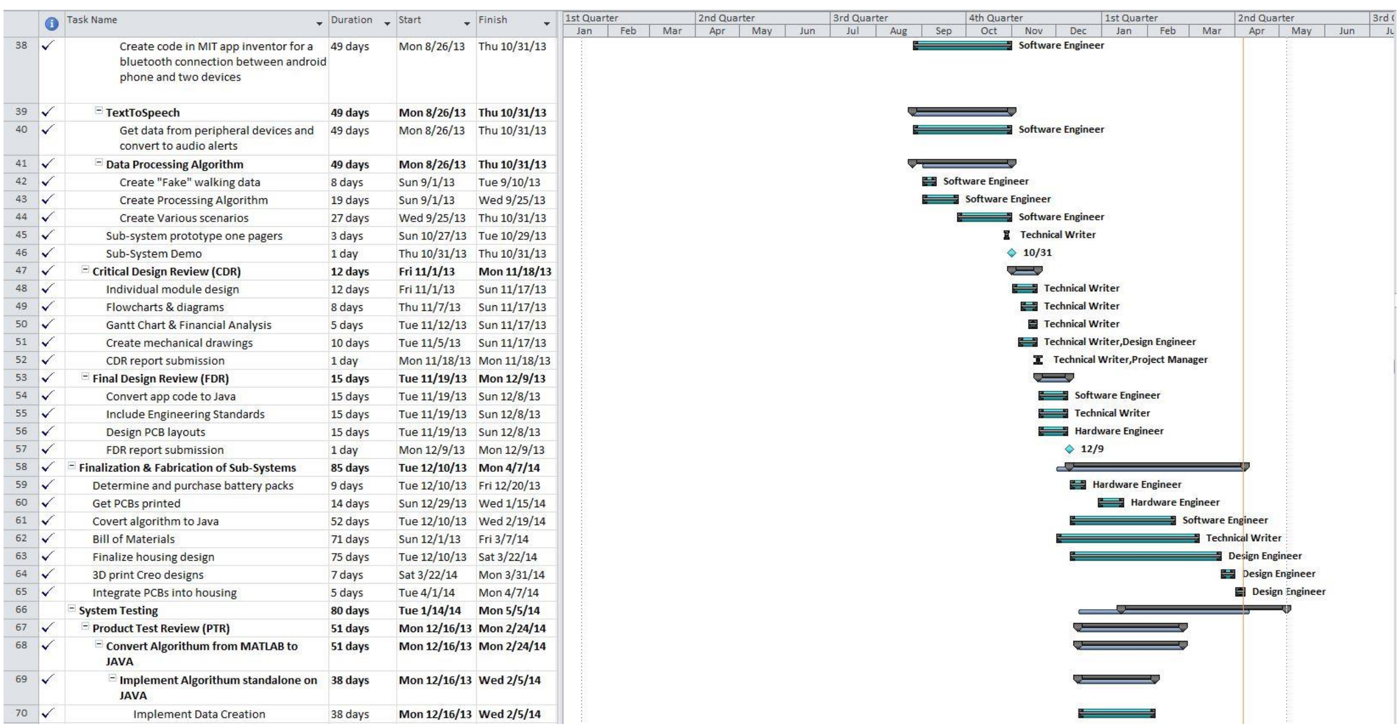
Figure 15.13: Text-to-speech notification generator module execution flowchart.

Appendix C: Gantt Chart



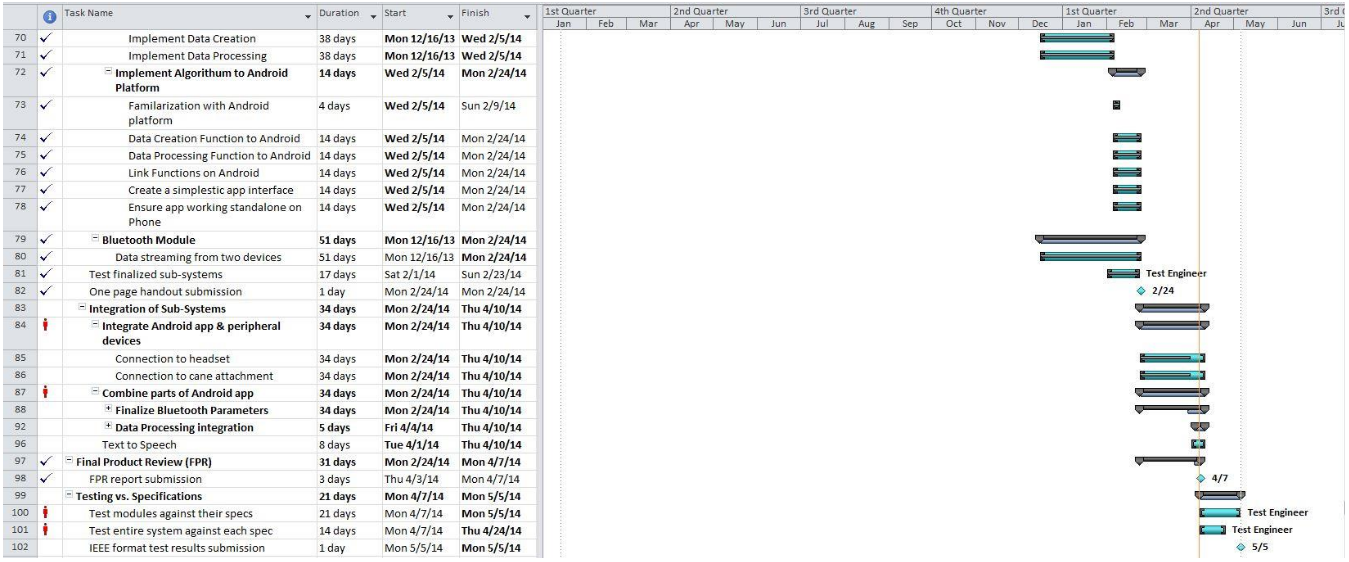
		Task Name	Duration	Start	Finish
1	✓	**Research & Idea Development**	**11.8 wks**	**Mon 1/14/13**	**Wed 4/3/13**
2	✓	Facial recognition research	10 days	Tue 1/15/13	Sun 1/27/13
3	✓	Decide to create blind assistive device	1 day	Mon 1/28/13	Mon 1/28/13
4	✓	Research existing assistive devices	5 days	Wed 1/30/13	Tue 2/5/13
5	✓	Requirements Draft	6 days	Fri 2/1/13	Fri 2/15/13
6	✓	Specifications Draft	13 days	Wed 2/27/13	Fri 3/15/13
7	✓	Decide on collision avoidance only	1 day	Wed 3/20/13	Wed 3/20/13
8	✓	Final requirements and specifications	11 days	Wed 3/20/13	Wed 4/3/13
9	✓	**Preliminary Design Development**	**27 days**	**Mon 4/1/13**	**Tue 5/7/13**
10	✓	**Proof of Principle (PoP)**	**23 days**	**Mon 4/1/13**	**Wed 5/1/13**
11	✓	**Hardware**	**19 days**	**Mon 4/1/13**	**Thu 4/25/13**
12	✓	Orded sensors	4 days	Mon 4/1/13	Thu 4/4/13
13	✓	ATMega16 coding	16 days	Thu 4/4/13	Thu 4/25/13
14	✓	Get data from sonar sensor	1 day	Fri 4/5/13	Fri 4/5/13
15	✓	Get data from accelerometer	1 day	Fri 4/5/13	Fri 4/5/13
16	✓	Test IR sensor capabilities	7 days	Sat 4/6/13	Sat 4/13/13
17	✓	Implement circuitry	15 days	Fri 4/5/13	Thu 4/25/13
18	✓	**Android Application**	**19 days**	**Mon 4/1/13**	**Thu 4/25/13**
19	✓	Research of bluetooth technology implementations	19 days	Mon 4/1/13	Thu 4/25/13
20	✓	Learned how to code in java	19 days	Mon 4/1/13	Thu 4/25/13
21	✓	PDR and PoP report submission	8 days	Thu 4/18/13	Mon 4/29/13
22	✓	PDR and PoP presentation	1 day	Wed 5/1/13	Wed 5/1/13
23	✓	**Low-Level Detailed Design**	**76 days**	**Mon 8/26/13**	**Mon 12/9/13**
24	✓	**Sub-System Design**	**42 days**	**Wed 9/4/13**	**Thu 10/31/13**
25	✓	Define scope of sub-system demos	2 wks	Thu 9/5/13	Wed 9/18/13
26	✓	Sub-System Prototype Plan	8 days	Thu 9/19/13	Mon 9/30/13
27	✓	**Headset & Cane Attachment**	**6.2 wks**	**Thu 9/19/13**	**Thu 10/31/13**
28	✓	Research AVR Microcontrollers	4 days	Thu 9/19/13	Tue 9/24/13
29	✓	Design both circuits	13 days	Tue 9/24/13	Thu 10/10/13
30	✓	Order (2) ATtiny1634s, (2) 4MHz crystal oscillators, and (2) Bluetooth modules	4 days	Fri 10/4/13	Wed 10/9/13
31	✓	Order 20-pin SOIP to DIP boards	4 days	Mon 10/14/13	Thu 10/17/13
32	✓	Solder ATtiny1634 to boards	2 days	Fri 10/18/13	Sat 10/19/13
33	✓	Build circuits on breadboards	3 days	Sun 10/20/13	Tue 10/22/13
34	✓	Test circuits	7 days	Tue 10/22/13	Wed 10/30/13
35	✓	**Android Application**	**49 days**	**Mon 8/26/13**	**Thu 10/31/13**
36	✓	**Bluetooth**	**49 days**	**Mon 8/26/13**	**Thu 10/31/13**
37	✓	Send data over bluetooth connection	49 days	Mon 8/26/13	Thu 10/31/13

Figure 15.14: Full Gantt chart (Part 1).



		Task Name	Duration	Start	Finish
38	✓	Create code in MIT app inventor for a bluetooth connection between android phone and two devices	49 days	Mon 8/26/13	Thu 10/31/13
39	✓	**TextToSpeech**	**49 days**	**Mon 8/26/13**	**Thu 10/31/13**
40	✓	Get data from peripheral devices and convert to audio alerts	49 days	Mon 8/26/13	Thu 10/31/13
41	✓	**Data Processing Algorithm**	**49 days**	**Mon 8/26/13**	**Thu 10/31/13**
42	✓	Create "Fake" walking data	8 days	Sun 9/1/13	Tue 9/10/13
43	✓	Create Processing Algorithm	19 days	Sun 9/1/13	Wed 9/25/13
44	✓	Create Various scenarios	27 days	Wed 9/25/13	Thu 10/31/13
45	✓	Sub-system prototype one pagers	3 days	Sun 10/27/13	Tue 10/29/13
46	✓	Sub-System Demo	1 day	Thu 10/31/13	Thu 10/31/13
47	✓	**Critical Design Review (CDR)**	**12 days**	**Fri 11/1/13**	**Mon 11/18/13**
48	✓	Individual module design	12 days	Fri 11/1/13	Sun 11/17/13
49	✓	Flowcharts & diagrams	8 days	Thu 11/7/13	Sun 11/17/13
50	✓	Gantt Chart & Financial Analysis	5 days	Tue 11/12/13	Sun 11/17/13
51	✓	Create mechanical drawings	10 days	Tue 11/5/13	Sun 11/17/13
52	✓	CDR report submission	1 day	Mon 11/18/13	Mon 11/18/13
53	✓	**Final Design Review (FDR)**	**15 days**	**Tue 11/19/13**	**Mon 12/9/13**
54	✓	Convert app code to Java	15 days	Tue 11/19/13	Sun 12/8/13
55	✓	Include Engineering Standards	15 days	Tue 11/19/13	Sun 12/8/13
56	✓	Design PCB layouts	15 days	Tue 11/19/13	Sun 12/8/13
57	✓	FDR report submission	1 day	Mon 12/9/13	Mon 12/9/13
58	✓	**Finalization & Fabrication of Sub-Systems**	**85 days**	**Tue 12/10/13**	**Mon 4/7/14**
59	✓	Determine and purchase battery packs	9 days	Tue 12/10/13	Fri 12/20/13
60	✓	Get PCBs printed	14 days	Sun 12/29/13	Wed 1/15/14
61	✓	Covert algorithm to Java	52 days	Tue 12/10/13	Wed 2/19/14
62	✓	Bill of Materials	71 days	Sun 12/1/13	Fri 3/7/14
63	✓	Finalize housing design	75 days	Tue 12/10/13	Sat 3/22/14
64	✓	3D print Creo designs	7 days	Sat 3/22/14	Mon 3/31/14
65	✓	Integrate PCBs into housing	5 days	Tue 4/1/14	Mon 4/7/14
66		**System Testing**	**80 days**	**Tue 1/14/14**	**Mon 5/5/14**
67	✓	**Product Test Review (PTR)**	**51 days**	**Mon 12/16/13**	**Mon 2/24/14**
68	✓	**Convert Algorithm from MATLAB to JAVA**	**51 days**	**Mon 12/16/13**	**Mon 2/24/14**
69	✓	**Implement Algorithm standalone on JAVA**	**38 days**	**Mon 12/16/13**	**Wed 2/5/14**
70	✓	Implement Data Creation	38 days	Mon 12/16/13	Wed 2/5/14

Figure 15.15: Full Gantt chart (Part 2).



ID		Task Name	Duration	Start	Finish
70	✓	Implement Data Creation	38 days	Mon 12/16/13	Wed 2/5/14
71	✓	Implement Data Processing	38 days	Mon 12/16/13	Wed 2/5/14
72	✓	Implement Algorithm to Android Platform	14 days	Wed 2/5/14	Mon 2/24/14
73	✓	Familarization with Android platform	4 days	Wed 2/5/14	Sun 2/9/14
74	✓	Data Creation Function to Android	14 days	Wed 2/5/14	Mon 2/24/14
75	✓	Data Processing Function to Android	14 days	Wed 2/5/14	Mon 2/24/14
76	✓	Link Functions on Android	14 days	Wed 2/5/14	Mon 2/24/14
77	✓	Create a simplestic app interface	14 days	Wed 2/5/14	Mon 2/24/14
78	✓	Ensure app working standalone on Phone	14 days	Wed 2/5/14	Mon 2/24/14
79	✓	Bluetooth Module	51 days	Mon 12/16/13	Mon 2/24/14
80	✓	Data streaming from two devices	51 days	Mon 12/16/13	Mon 2/24/14
81	✓	Test finalized sub-systems	17 days	Sat 2/1/14	Sun 2/23/14
82	✓	One page handout submission	1 day	Mon 2/24/14	Mon 2/24/14
83		Integration of Sub-Systems	34 days	Mon 2/24/14	Thu 4/10/14
84		Integrate Android app & peripheral devices	34 days	Mon 2/24/14	Thu 4/10/14
85		Connection to headset	34 days	Mon 2/24/14	Thu 4/10/14
86		Connection to cane attachment	34 days	Mon 2/24/14	Thu 4/10/14
87		Combine parts of Android app	34 days	Mon 2/24/14	Thu 4/10/14
88		Finalize Bluetooth Parameters	34 days	Mon 2/24/14	Thu 4/10/14
92		Data Processing integration	5 days	Fri 4/4/14	Thu 4/10/14
96		Text to Speech	8 days	Tue 4/1/14	Thu 4/10/14
97	✓	Final Product Review (FPR)	31 days	Mon 2/24/14	Mon 4/7/14
98	✓	FPR report submission	3 days	Thu 4/3/14	Mon 4/7/14
99		Testing vs. Specifications	21 days	Mon 4/7/14	Mon 5/5/14
100		Test modules against their specs	21 days	Mon 4/7/14	Mon 5/5/14
101		Test entire system against each spec	14 days	Mon 4/7/14	Thu 4/24/14
102		IEEE format test results submission	1 day	Mon 5/5/14	Mon 5/5/14

Figure 15.16: Full Gantt chart (Part 3).

Figure 15.17: Timeline with milestones from Gantt chart in Microsoft Project.

Appendix D: Economic Analysis Data

Week #	Labor Cost (per week - in $) Project Manager	Labor Cost (per week - in $) Design Engineer	Labor Cost (per week - in $) Hardware Engineer	Labor Cost (per week - in $) Software Engineer	Labor Cost (per week - in $) Test Engineer	Labor Cost (per week - in $) Technical Writer	Total Weekly Burn Rate Due to Labor	Total Labor Costs
1	$132.00	$171.00					$303.00	$303.00
2	$132.00	$171.00					$303.00	$606.00
3	$132.00	$171.00					$303.00	$909.00
4	$132.00						$132.00	$1,041.00
5	$132.00					$30.00	$162.00	$1,203.00
6	$132.00					$30.00	$162.00	$1,365.00
7	$132.00					$30.00	$162.00	$1,527.00
8	$132.00					$30.00	$162.00	$1,689.00
9	$132.00					$30.00	$162.00	$1,851.00
10	$132.00					$30.00	$162.00	$2,013.00
11	$132.00					$30.00	$162.00	$2,175.00
12	$132.00					$30.00	$162.00	$2,337.00
13	$132.00		$360.00	$200.00		$150.00	$842.00	$3,179.00
14	$132.00		$360.00	$200.00		$150.00	$842.00	$4,021.00
15	$132.00		$360.00	$200.00		$150.00	$842.00	$4,863.00
16	$132.00		$360.00	$200.00		$150.00	$842.00	$5,705.00
17	$132.00						$132.00	$5,837.00
18	$132.00						$132.00	$5,969.00
19	$132.00					$90.00	$222.00	$6,191.00
20	$132.00						$132.00	$6,323.00
21	$132.00		$240.00	$400.00	$36.00		$808.00	$7,131.00
22	$132.00		$240.00	$400.00	$36.00		$808.00	$7,939.00
23	$132.00		$240.00	$400.00	$36.00		$808.00	$8,747.00
24	$132.00		$240.00	$400.00	$36.00		$808.00	$9,555.00
25	$132.00		$240.00	$400.00	$36.00	$60.00	$868.00	$10,423.00
26	$132.00		$240.00	$400.00	$36.00		$808.00	$11,231.00
27	$132.00	$57				$450	$639.00	$11,870.00
28	$132.00	$57				$450	$639.00	$12,509.00
29	$132.00		$384	$400		$60	$976.00	$13,485.00
30	$132.00		$384	$400		$60	$976.00	$14,461.00
31	$132.00		$384	$400		$60	$976.00	$15,437.00
32	$132.00	$171	$48	$40			$391.00	$15,828.00
33	$132.00	$171	$48	$40			$391.00	$16,219.00
34	$132.00	$171	$48	$40			$391.00	$16,610.00
35	$132.00	$171	$48	$40			$391.00	$17,001.00
36	$132.00	$171	$48	$40			$391.00	$17,392.00
37	$132.00	$171	$48	$40			$391.00	$17,783.00
38	$132.00	$171	$48	$40			$391.00	$18,174.00
39	$132.00		$48	$120		$60	$360.00	$18,534.00
40	$132.00		$48	$120	$72	$60	$432.00	$18,966.00
41	$132.00		$48	$120	$72	$60	$432.00	$19,398.00
42	$132.00		$48	$120	$72	$60	$432.00	$19,830.00
43	$132.00		$48	$120	$72	$60	$432.00	$20,262.00
44	$132.00		$48	$120	$72	$60	$432.00	$20,694.00
45	$132.00		$48	$120	$72	$60	$432.00	$21,126.00
46	$132.00		$48	$120	$72	$60	$432.00	$21,558.00
47	$132.00				$72	$60	$264.00	$21,822.00
48	$132.00				$72	$60	$264.00	$22,086.00
49	$132.00				$72	$60	$264.00	$22,350.00
50	$132.00				$72	$60	$264.00	$22,614.00
Totals	$6,600	$1,824	$4,752	$5,640	$1,008	$2,790		

"=Actual data up to week #46

"=Estimated data for week #47 to week #50

"=PROJECTED cumulative spending to end of project

Table 15.1: Total labor costs spreadsheet for Labor Costs vs. Time Graph.

Appendix E: Module Matrix

Project Phase	Estimated Hours Left	Actual Hours	Percent Completed
Headset & Cane Attachment Sub-Systems	**0**	**576**	**100%**
Ultrasonic & Accelerometer Modules	0	148	100%
Preliminary Research	0	10	100%
Design	0	15	100%
Implementation	0	40	100%
Testing	0	45	100%
Error Correction	0	38	100%
Microcontroller Coding	0	145	100%
Preliminary Research	0	10	100%
Design	0	10	100%
Implementation	0	55	100%
Testing	0	40	100%
Error Correction	0	30	100%
Bluetooth Transceiver Interfacing	0	96	100%
Preliminary Research	0	10	100%
Design	0	10	100%
Implementation	0	26	100%
Testing	0	25	100%
Error Correction	0	25	100%
Battery & Power Management Design	0	77	100%
Preliminary Research	0	15	100%
Design	0	25	100%
Implementation	0	30	100%
Testing	0	5	100%
Error Correction	0	2	100%
Printed Circuit Board Design	0	110	100%
Preliminary Research	0	15	100%
Design	0	50	100%
Implementation	0	20	100%
Testing	0	10	100%
Error Correction	0	15	100%
Android Application Sub-Systems	**157**	**473**	**75%**
Bluetooth Module	90	145	62%
Connection between three phones	0	35	100%
Connection between phone and headset/cane	15	10	40%
Integration of code with algorithm JAVA version	40	0	0%
Testing	10	50	83%
Implementation	10	10	50%

Error Correction	15	40	73%
Data Processing (Algorithm) Module	55	265	83%
"Fake" Data creator module	0	65	100%
Algorithm MATLAB version	0	50	100%
Algorithm JAVA version	0	40	100%
Testing	15	50	77%
Implementation	20	25	56%
Error Correction	20	35	64%
Text-to-Speech Notification Generator	12	63	84%
Coding	2	35	95%
Testing	5	15	75%
Implementation	2	3	60%
Error Correction	3	10	77%

Table 15.2: Module Matrix.

Appendix F: Board Fabrication Details

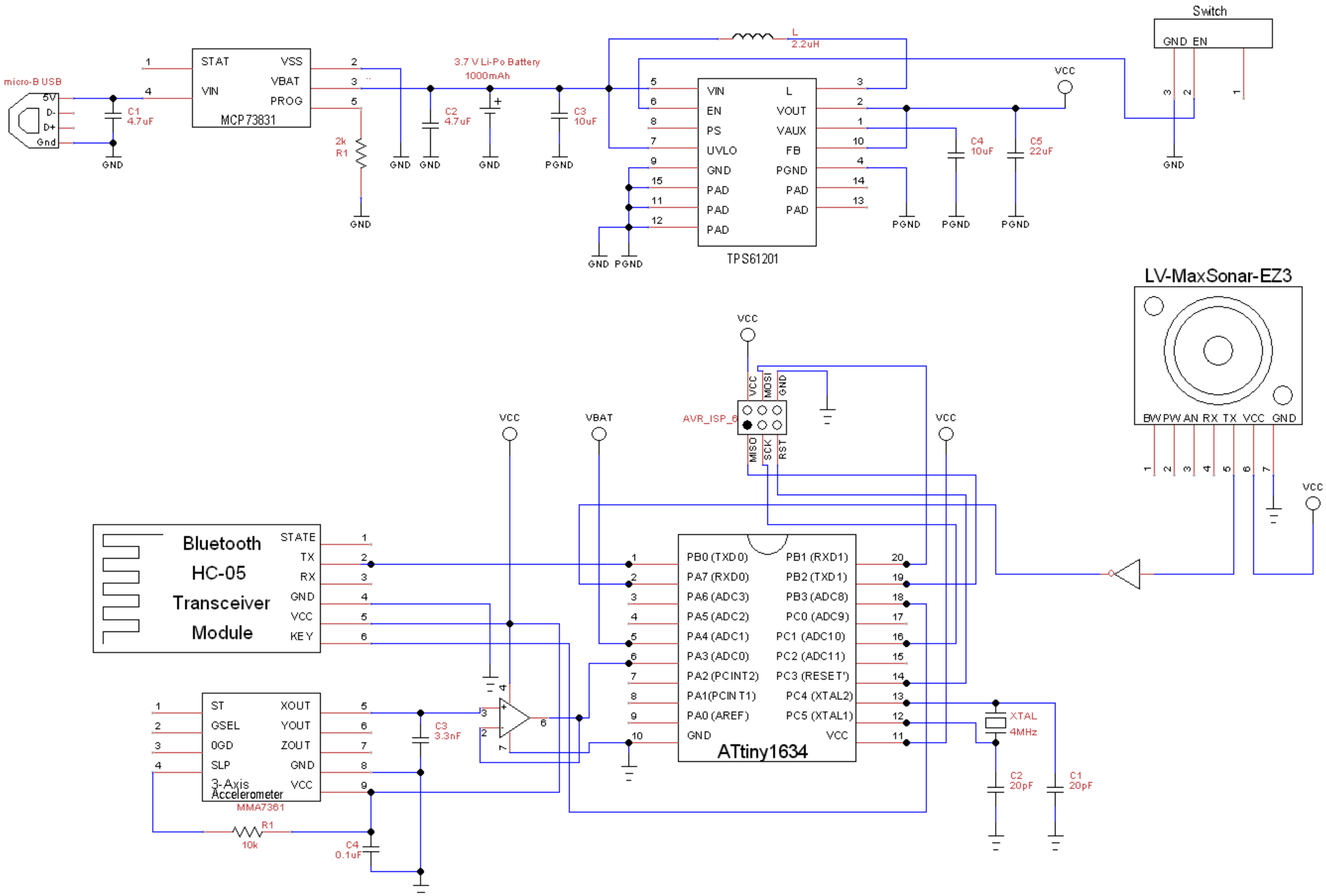


Figure 15.18: Schematic for headset and cane attachment sub-systems.

Quantity	Description of the Part	Manufacturer	Manufacturer's Part Number	Supplier	Supplier's Catalog Number	Cost	Total Cost
2	ATtiny1634 8-bit MCU	Atmel	ATTINY1634-SU	Mouser	556-ATTINY1634-SU	$1.76	$3.52
2	3.7V Li-Ion battery - 1000mAh	Unionfortune	63450	Sparkfun	PRT-00339	$11.95	$23.90
2	LV-MaxSonar-EZ3	MaxBotix	MB1030	Sparkfun	SEN-08501	$22.36	$44.72
2	Three-axis low-g accelerometer	Freescale Semiconductor	MMA7361LCR1	Sparkfun	COM-09605	$7.95	$15.90
2	HC-05 Bluetooth transceiver	ITead Studio	HC-05	Ebay	HC-05	$9.28	$18.56
2	PowerCell Charger/Booster	Sparkfun	PRT-11231	Sparkfun	PRT-11231	$15.96	$31.92
2	SMD right-angle switch	On Shine Enterprise	KPS-1290	Sparkfun	COM-10860	$0.76	$1.52
2	Single Schmitt-Trigger inverter	Texas Instruments	SN74LVC1G14DBVR	Mouser	595-SN74LVC1G14DBVR	$0.49	$0.98
2	4MHz crystal oscillator	Fox	FOXSLF/040	Mouser	559-FOXS040-LF	$0.21	$0.42
2	Operational Amplifier	ST Microelectronics	TSV731	Mouser	511-TSV731ICT	$1.70	$3.40
4	20pF Capacitor	Vishay	VJ0402D200JXBAJ	Mouser	77-VJ0402D200JXBAJ	$0.30	$1.20
2	3.3nF Capacitor	Vishay	VJ0402Y332KXJCW1BC	Mouser	77-VJ0402Y332KXJCBC	$0.06	$0.12
2	0.1uF Capacitor	Murata Manufacturing	GRM155R71C104KA88D	Mouser	81-GRM155R71C104KA88	$0.10	$0.20
2	10kΩ Resistor	Vishay	TNPW060310K0DEEA	Mouser	71-TNPW060310K0DEEA	$0.09	$0.18
2	10MΩ Resistor	Vishay	MCT06030C1005FP500	Mouser	594-MCT06030C1005FP5	$0.07	$0.14
2	Printed Circuit Board	OSHPark		OSHPark		$7.95	$15.90
2	Mini Ball Bearings	Losi	LOSB1528	Amazon	B000KFW2C4	$1.52	$3.04
12	4-40 PCB Mounting Screws	TubeDepot	MS-ST-4-40	TubeDepot	MS-ST-4-40	$0.14	$1.40
						Total	$187.62

Table 15.3: Bill of Materials (BOM).

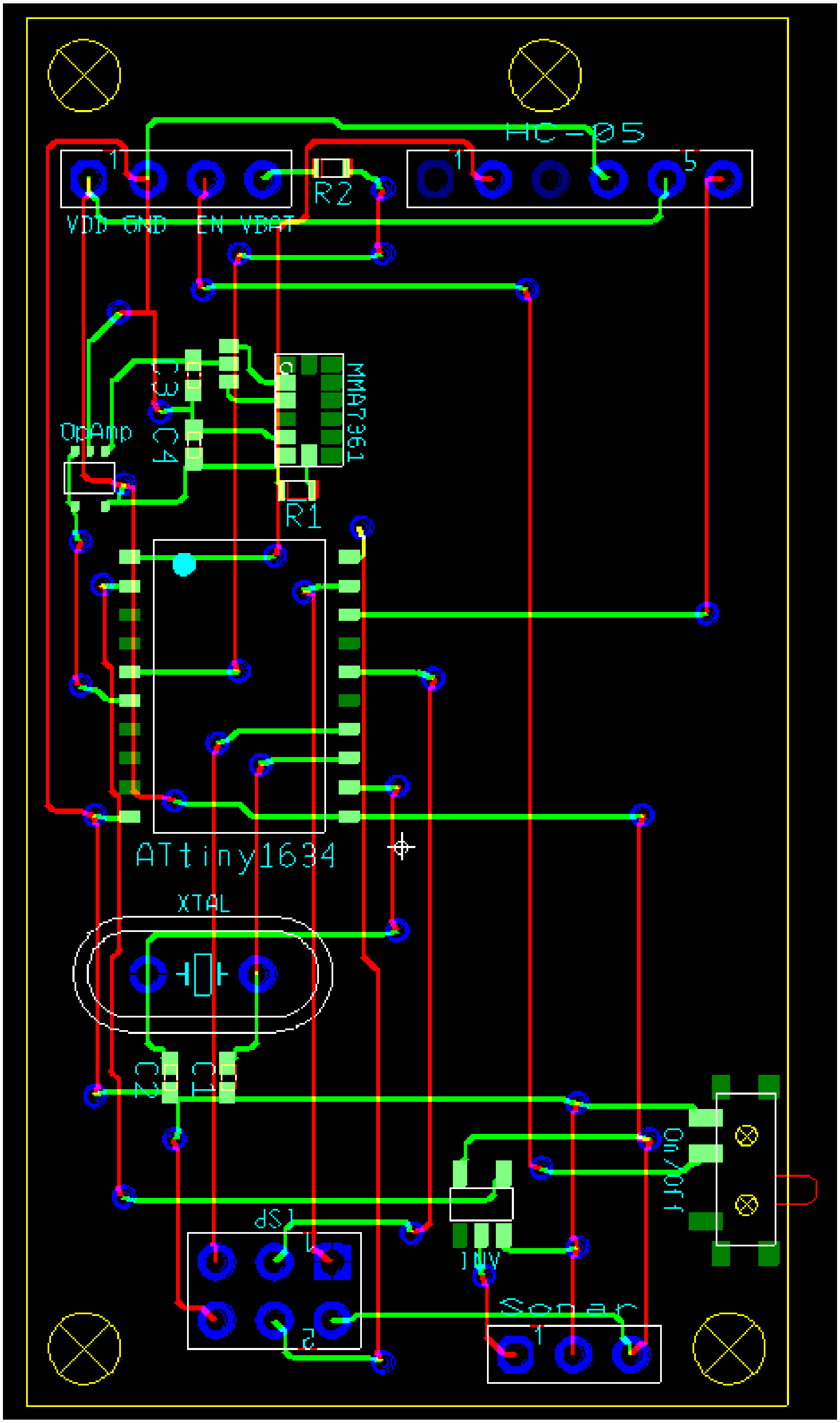


Figure 15.19: Printed Circuit Board layout.

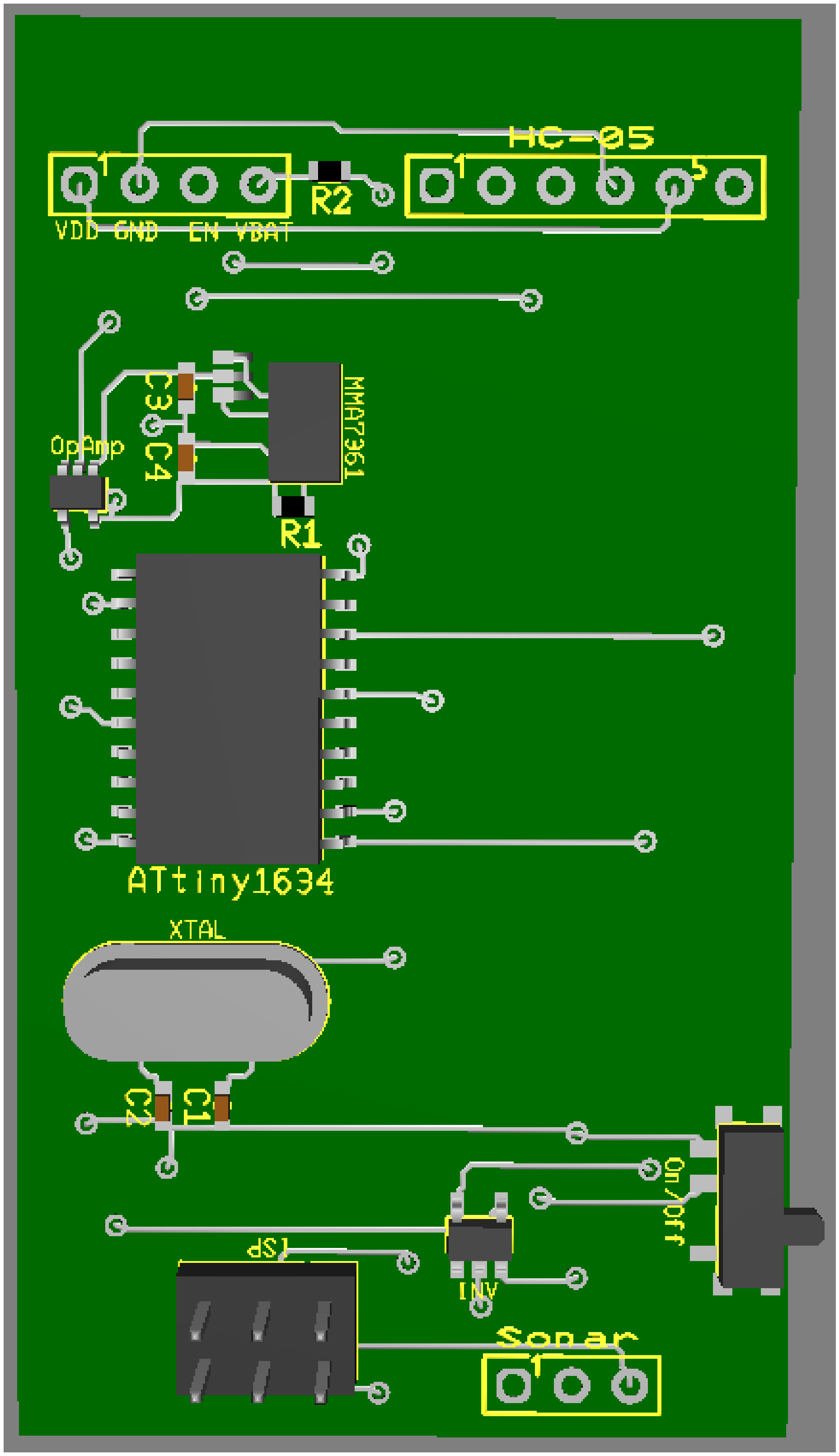


Figure 15.20: PCB 3D rendering.

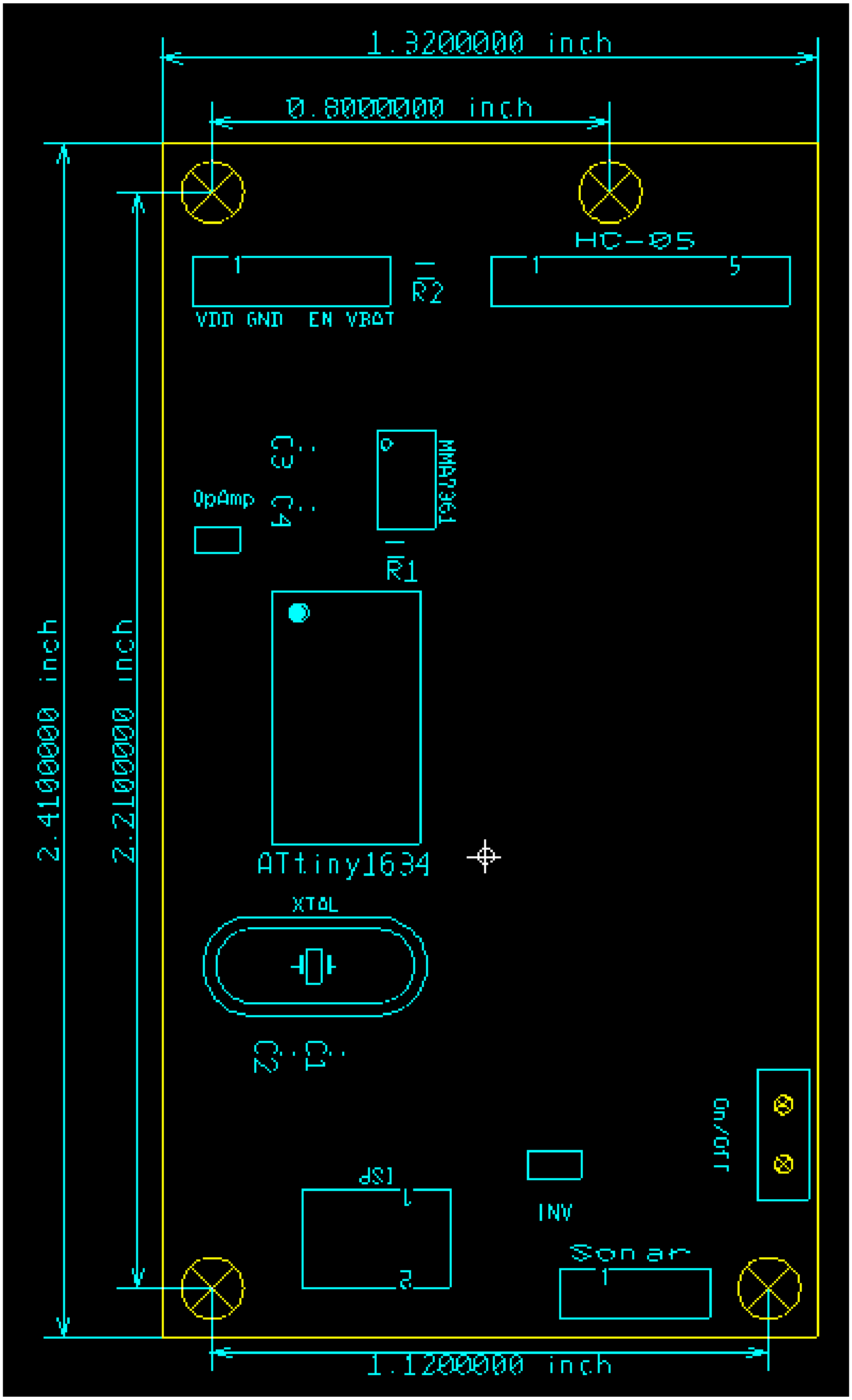


Figure 15.21: PCB outer dimensions.

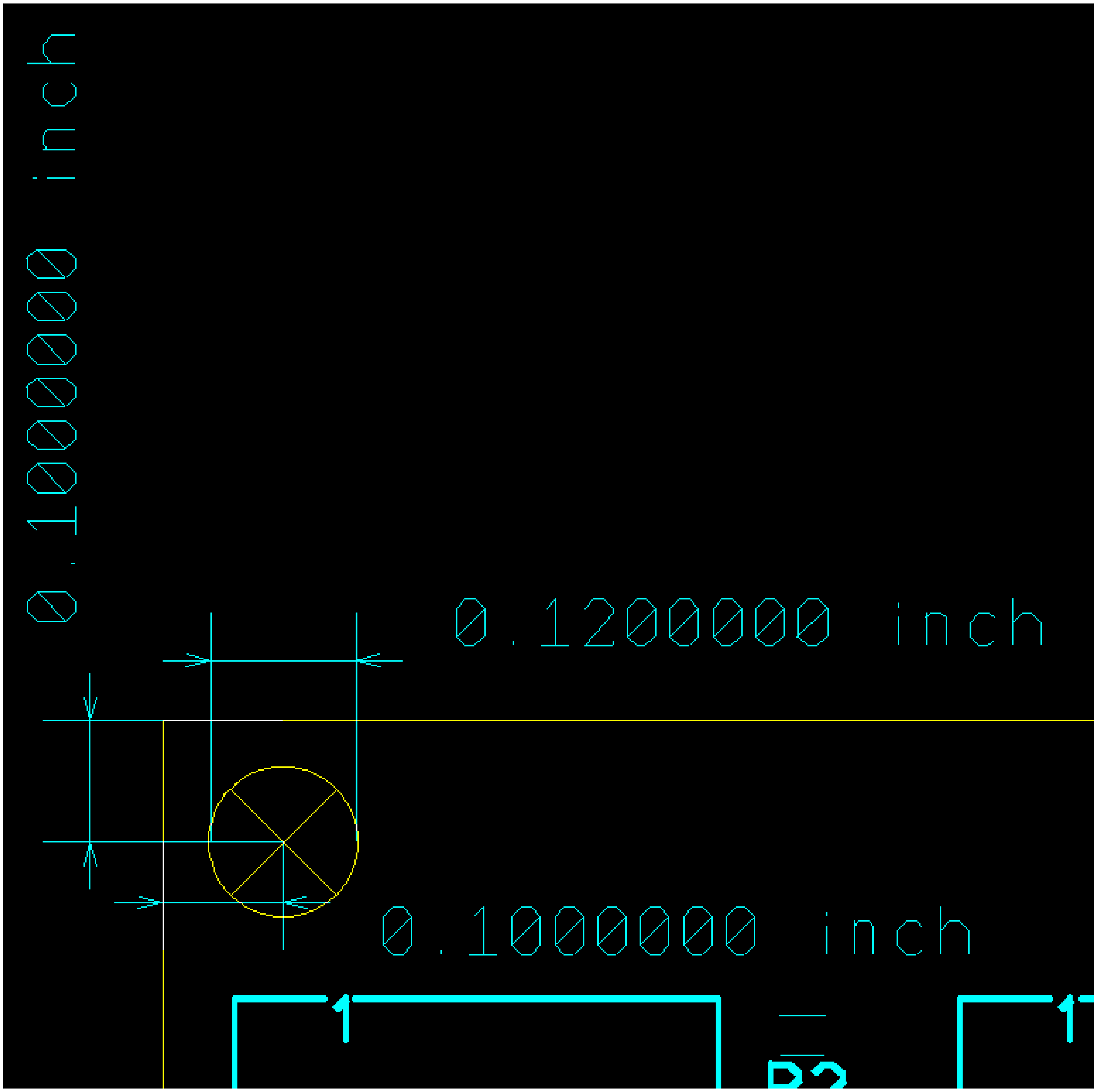


Figure 15.22: PCB drill hole dimensions.

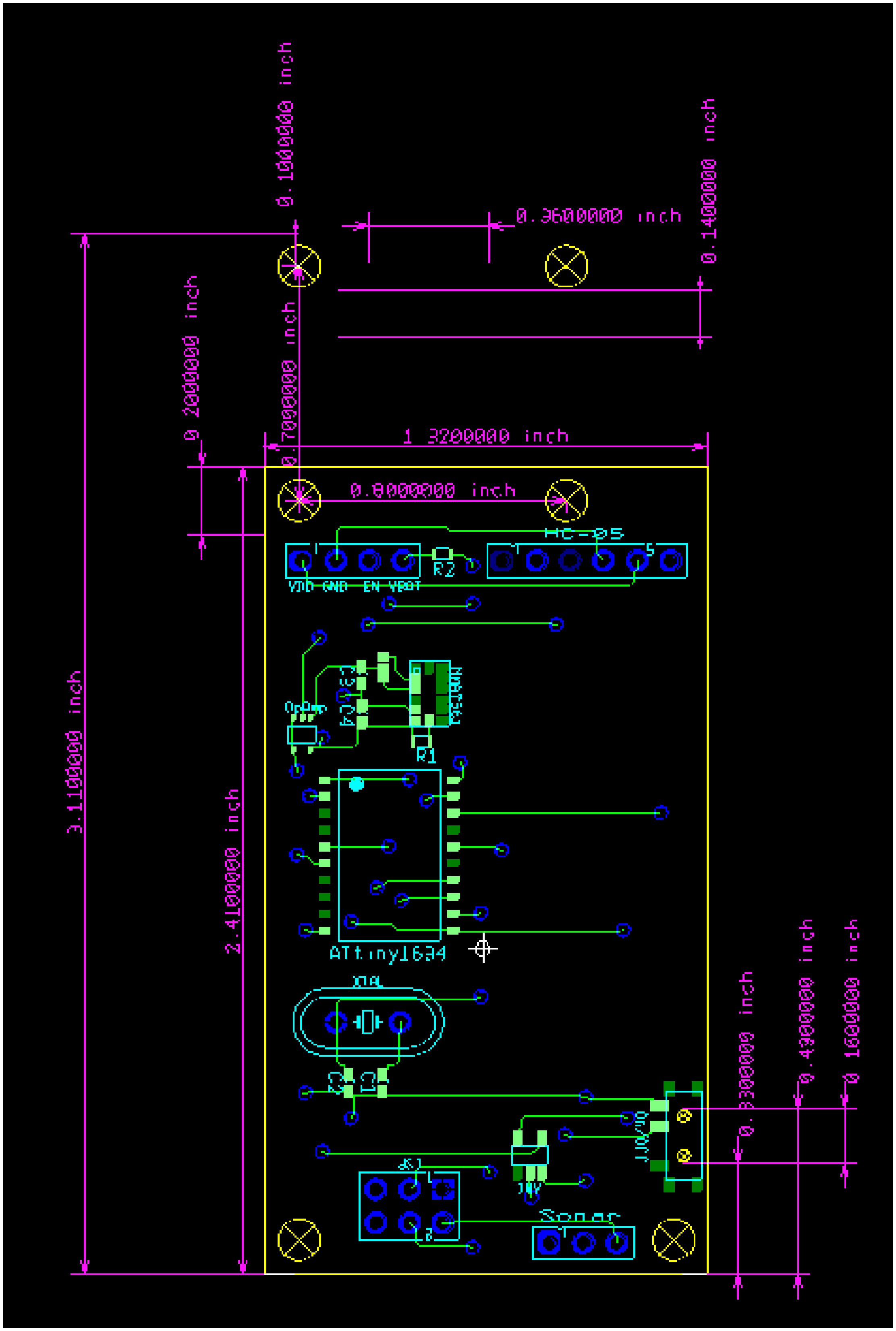


Figure 15.23: Entire PCB system dimensions.
Note: Additional two drill holes at top show how the PowerCell is to be mounted with this PCB.

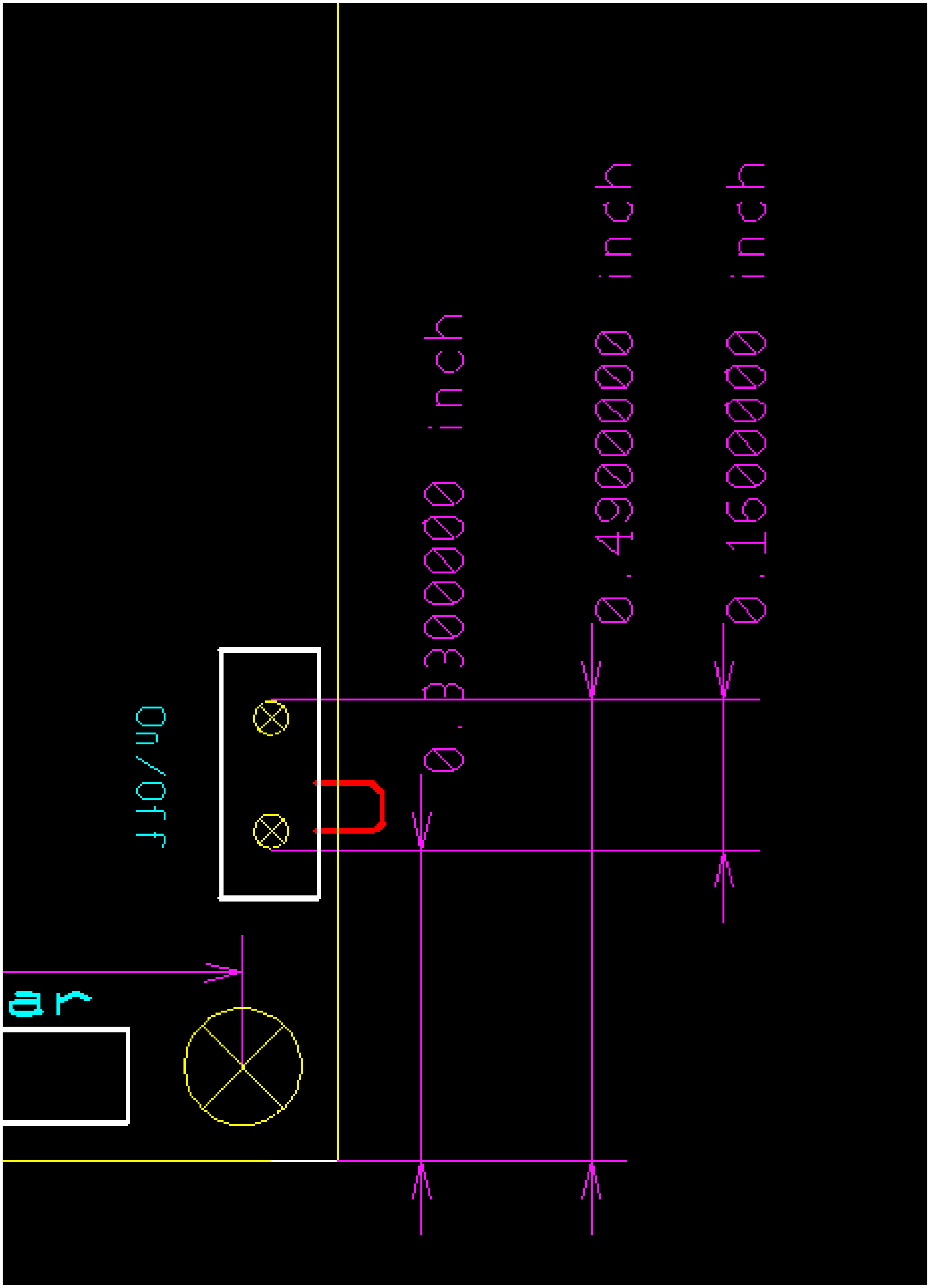


Figure 15.24: On/off switch location dimensions.

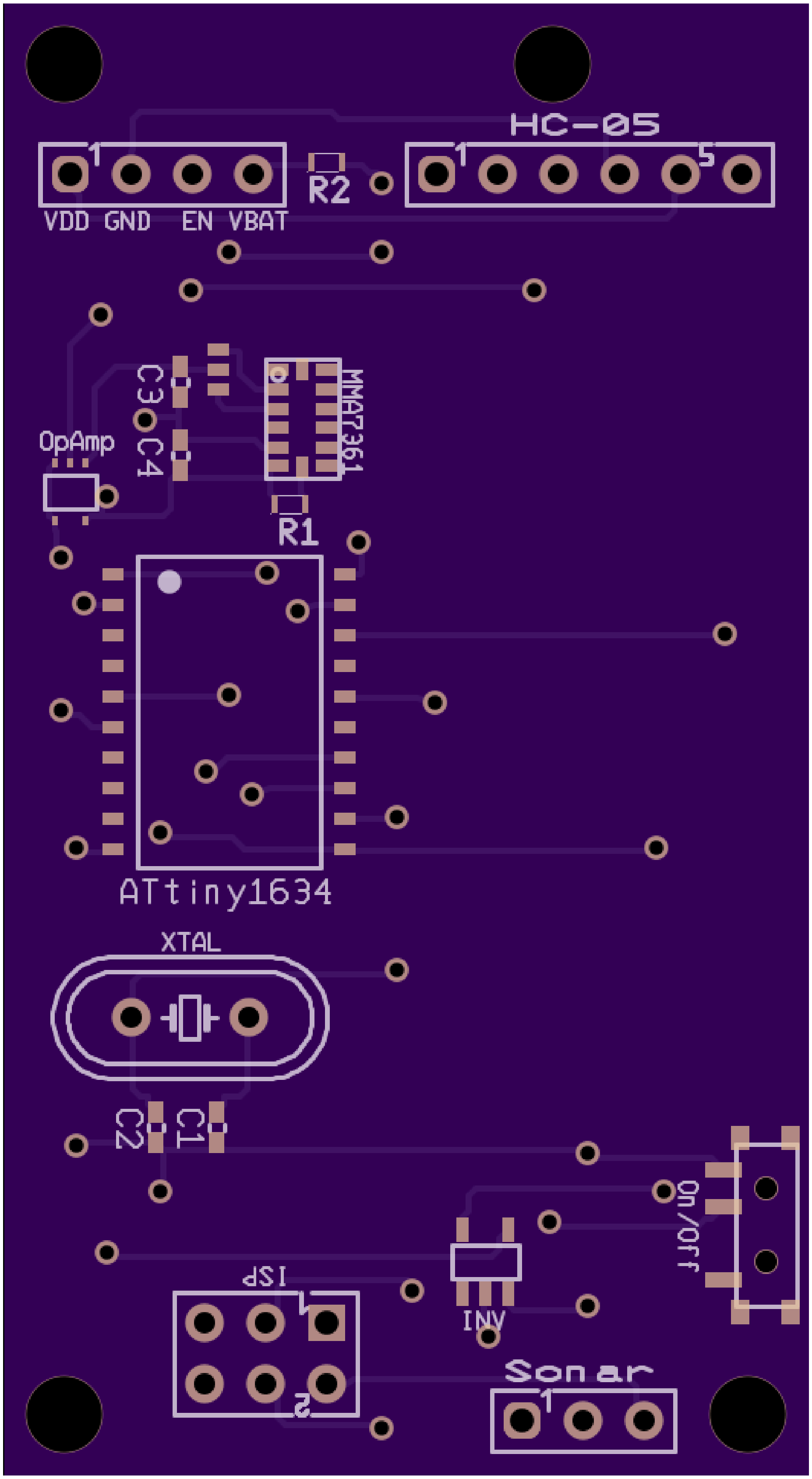


Figure 15.25: Printed PCB top rendering.

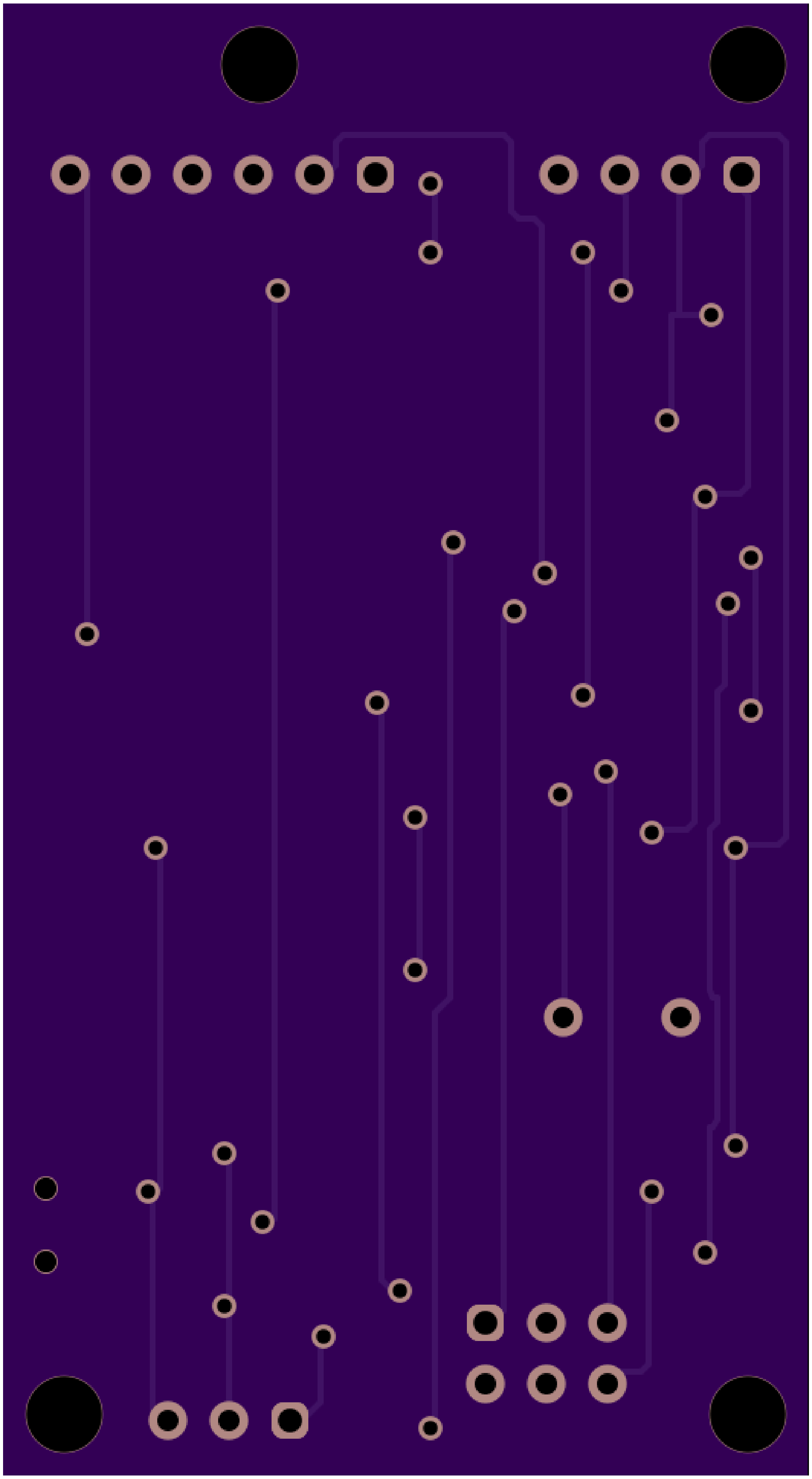


Figure 15.26: Printed PCB bottom rendering.

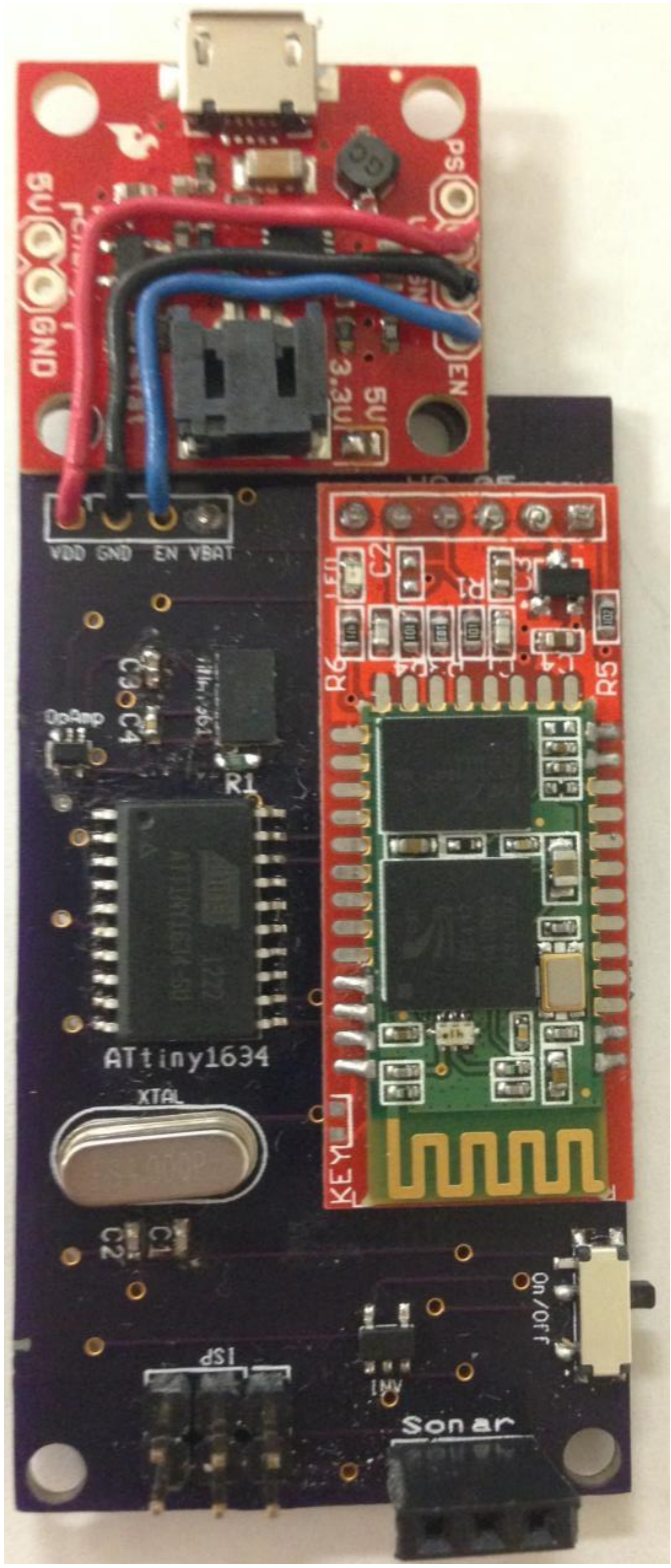


Figure 15.27: Final soldered headset circuit.

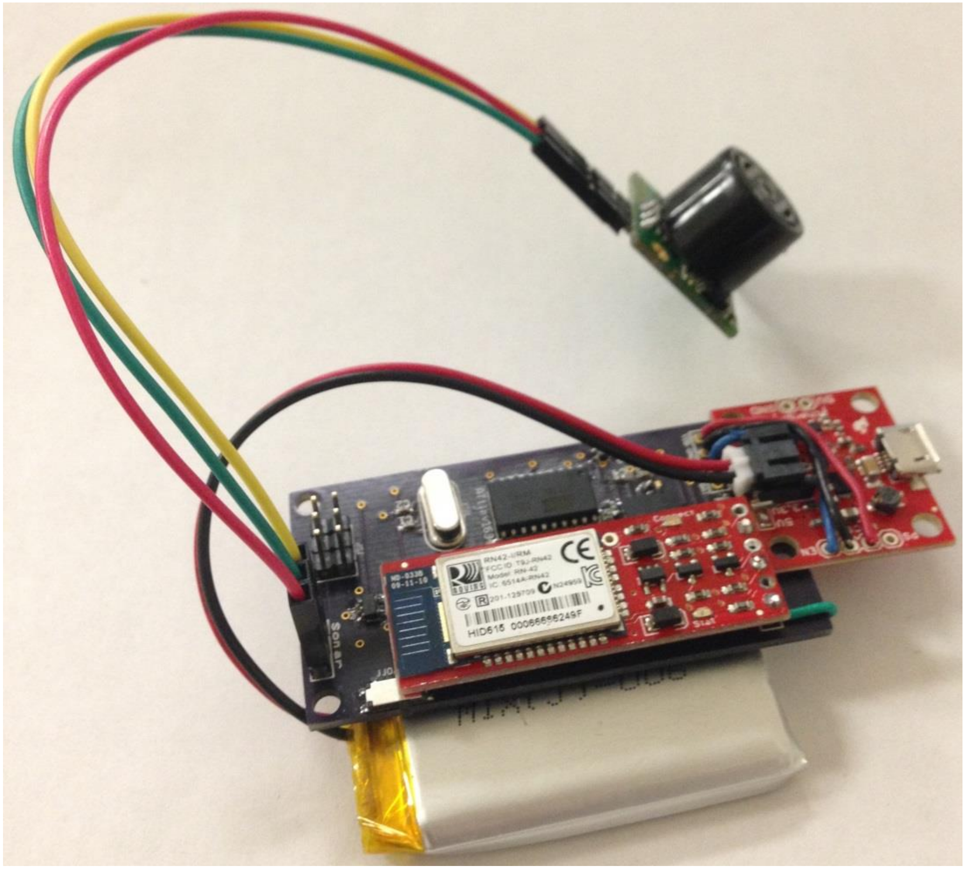


Figure 15.28: Entire circuit with battery and ultrasonic sensor.

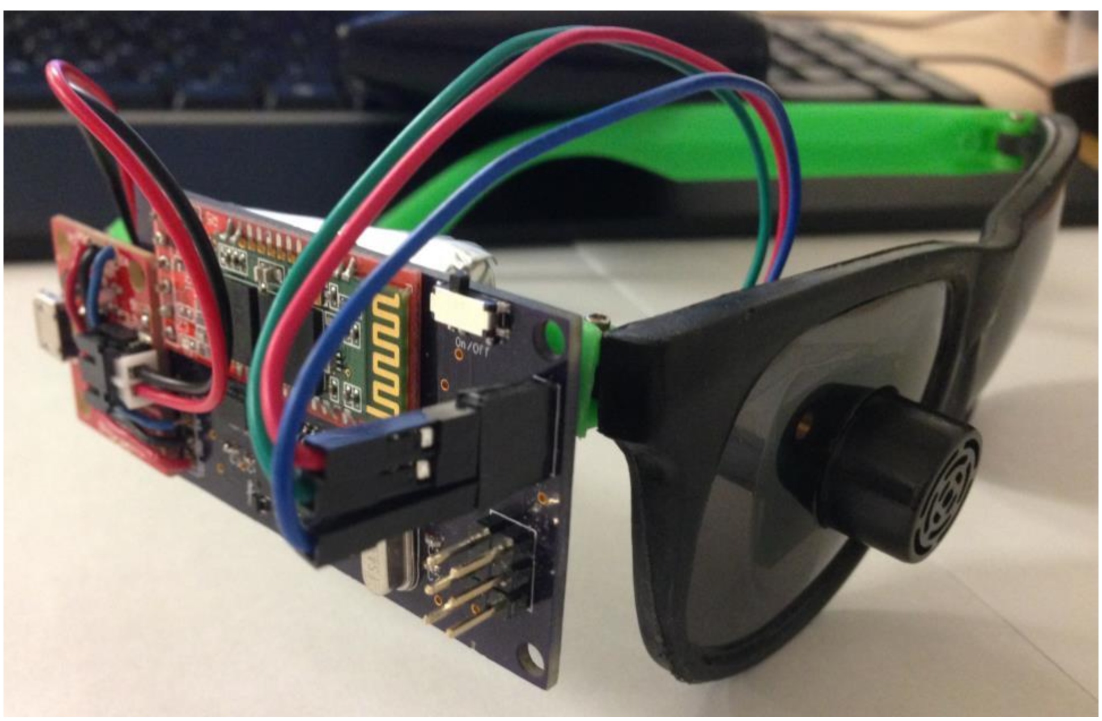


Figure 15.29: Prototype headset prior to housing addition.

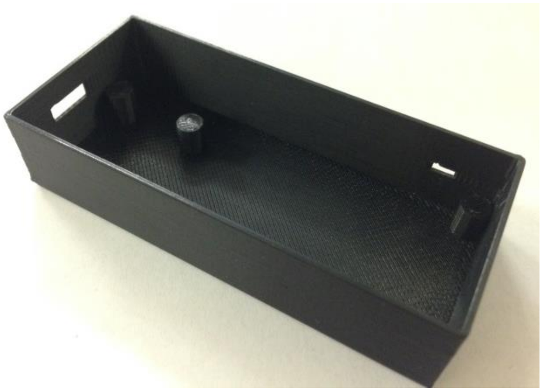


Figure 15.30: 3D-printed headset housing.

Figure 15.31: 3D-printed headset housing interior.

Figure 15.32: Battery fitted into headset housing.

Figure 15.33: Circuit fitted into headset housing.

Figure 15.34: Mini USB connection for charging.

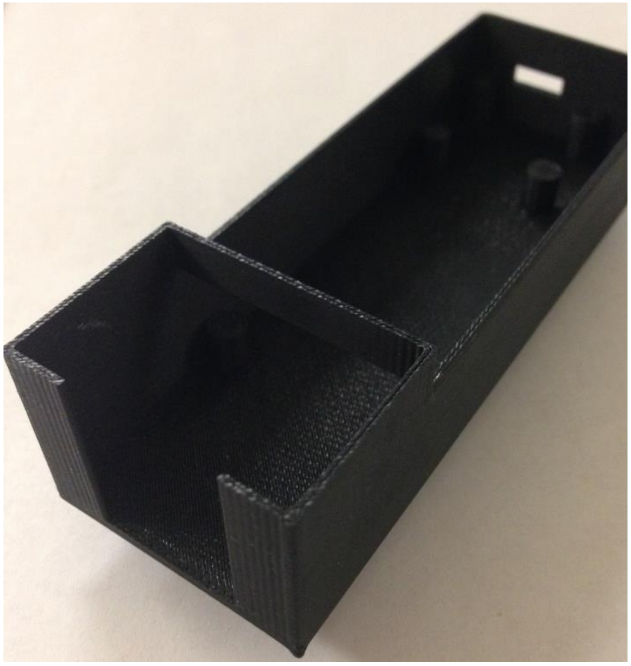


Figure 15.35: 3D-printed cane attachment housing.

Figure 15.36: Cane attachment housing inside.

Figure 15.37: Assembled headset housing with lid.

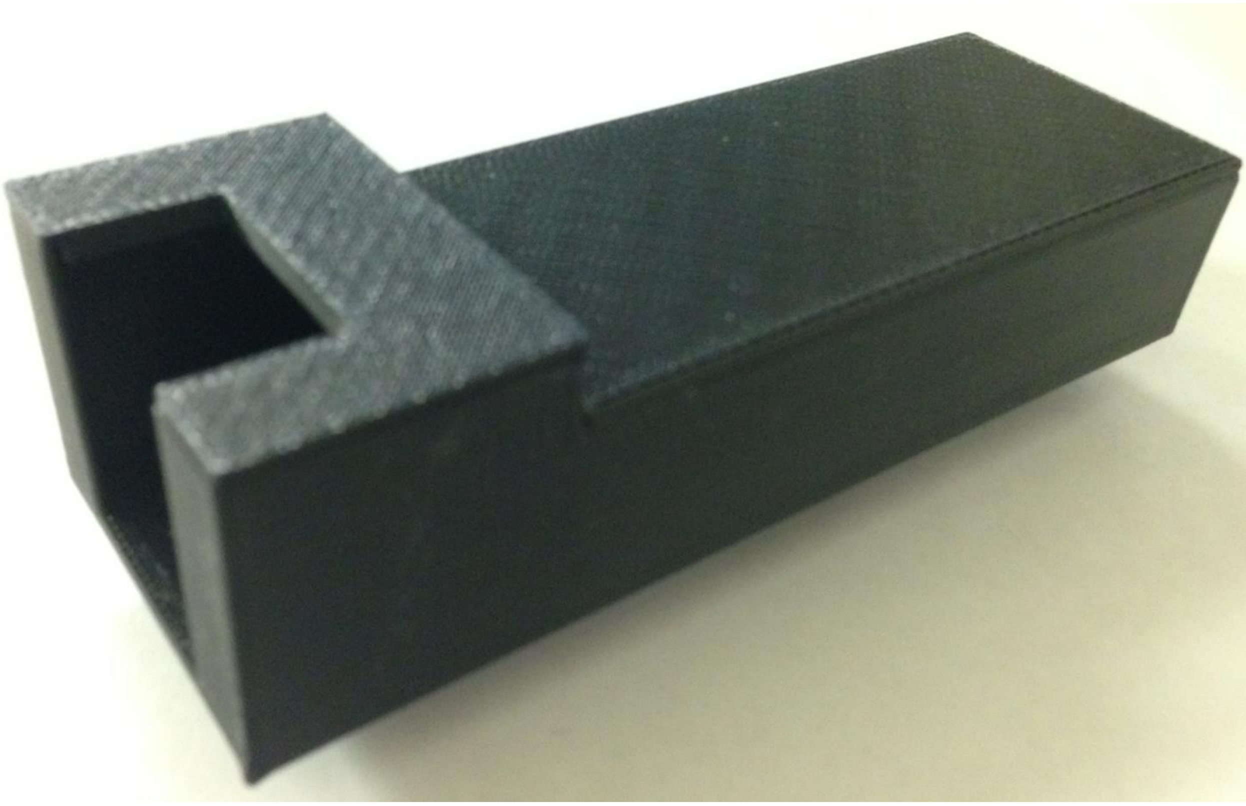


Figure 15.38: Assembled cane attachment housing with lids.

Appendix G: Test Results

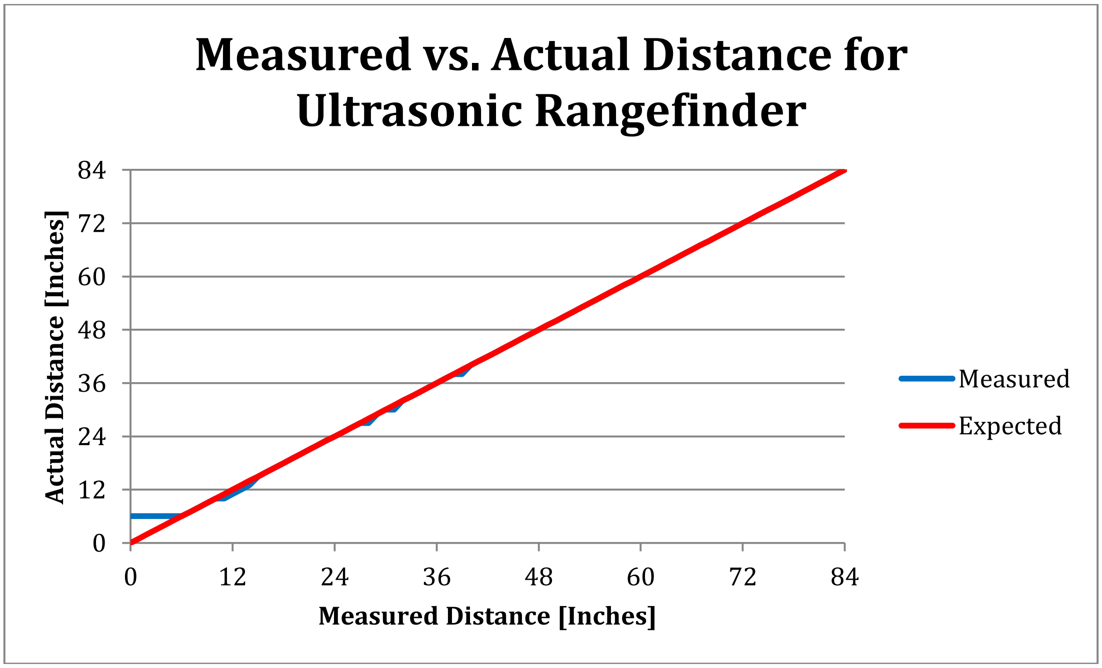


Figure 15.39: Ultrasonic rangefinder characteristic curve.

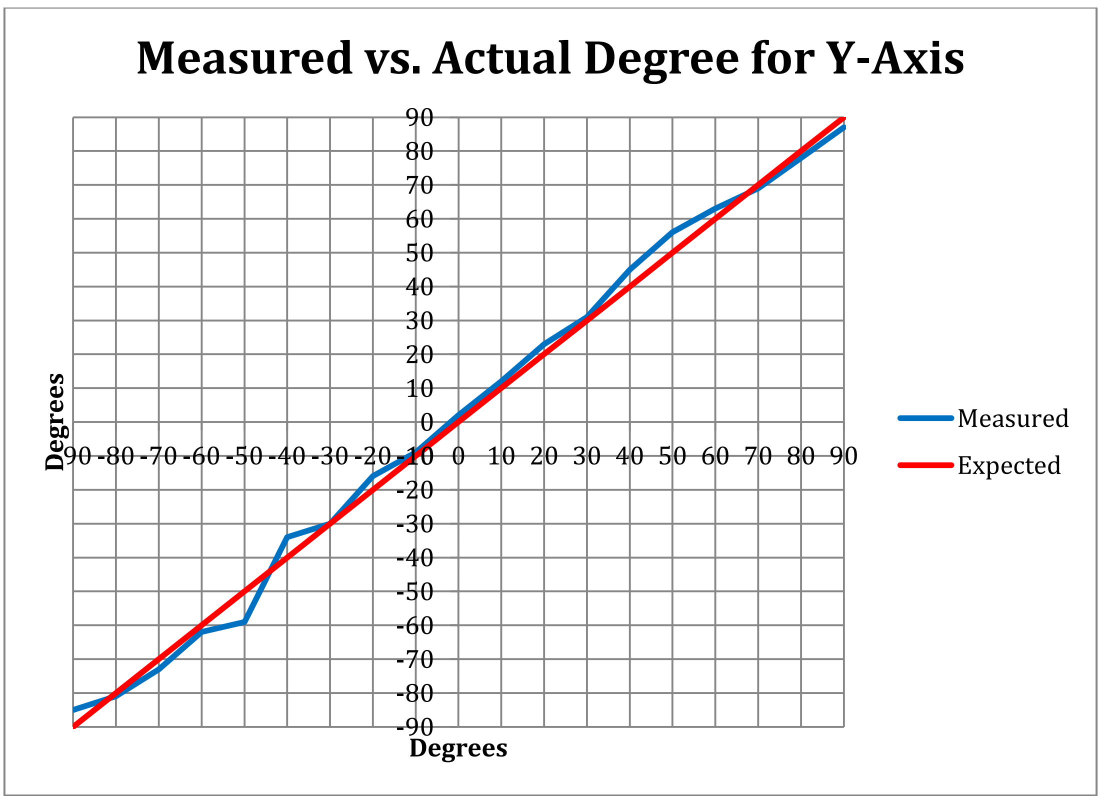


Figure 15.40: Accelerometer y-axis output characteristic curve.

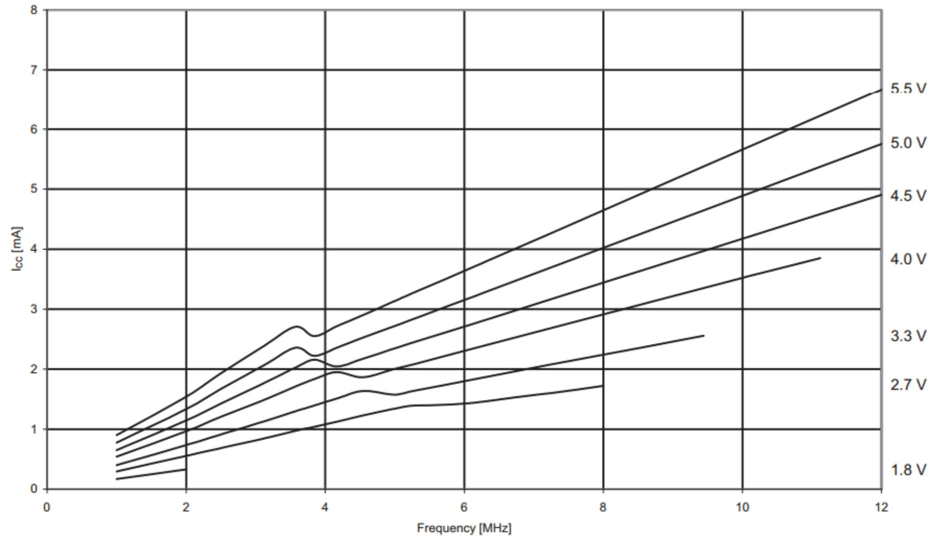


Figure 15.41: ATtiny1634 Active Supply Current vs. Frequency (1-12MHz)

Appendix H: Source Code

```c
/* Brandon Bernier
 * ECE 4925 Senior Design
 * Collision Avoidance System for the Visually Impaired
 * Headset Sub-System
 */

#define F_CPU 4000000  // 4MHz Clock
#define BAUD_VAL 25    // 9600 bps for 4MHz clock
#define LOW_BATT 600   // 20% battery left
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>

/* putchar into UDR0 */
int usart_putchar(char c, FILE *stream){
      if( c == '\n' ) usart_putchar('\r', stream);
      while( !(UCSR0A&(1<<UDRE0)) );             // Wait until UDR is empty
      UDR0 = c;
      return 0;
}

/* getchar from UDR0 */
int usart_getchar(FILE *stream){
      if( UCSR0A&(1<<RXC0) ) return UDR0;
      else return 0;
}

/* Redirect stdio stream to new functions */
FILE usart_str = FDEV_SETUP_STREAM(usart_putchar, usart_getchar, _FDEV_SETUP_RW);

/* Function Prototypes */
void initialize(void);
void start_adc(void);
unsigned char decode_number(unsigned char x);

/* Global Variables */
volatile unsigned char count=0, accel=1;
volatile unsigned char d=0, distance=0;
volatile int degree,vbatt,vbatt_old;
volatile int battery[5]={1024,1024,1024,1024,1024};

int main(void){
      initialize();
      while(1) start_adc();

      return 0;
}

/* USART0 Receive Interrupt Vector */
/* Program execution jumps here when USART0 receives data.
 * Serial data streams from the ultrasonic sensor one character
 * at a time in the form "R255\r" where R denotes the start of a
 * new reading, followed by 3 digits (distance in inches).
 * This section of code parses this input and creates an 8-bit
 * unsigned char holding the value in inches.
 */
ISR(USART0_RX_vect){
    unsigned char x;
    x = decode_number( getchar() );          // Gets the next character from the UDRE
    if( x == 'R' ){                          // Count variable keeps track of which
          count = 0;                         // digit we are looking for
          d = 0;
    }
    else if( x == '\r' ){}
    else{
          if( count == 1 )                   // Reading the hundreds place
          d += x*100;
          else if( count == 2)               // Reading the tens place
          d+= x*10;
          else if( count == 3){              // Reading the ones place
            d += x;
            distance = d;                            // Updates distance
            printf("HS%3d HA%4d\r",distance,degree);  // Send out new set of data
          }
    }
      count = (count+1)%5;
}
```

```
/* ADC Complete Interrupt Vector */
/* Program execution jumps here once the started ADC completes.
 * Combines lower 8 and upper 4 bits of the 10-bit ADC value.
 * Converts 10-bit ADC value to degrees.
 */
ISR(ADC_vect){
      unsigned char high,low;
      int adc=0;
      low = ADCL;
      high = ADCH;
      adc |= (high<<2)|low;

      if(accel){
        //printf("%d\r",adc);         // Print raw ADC value
        degree = (int)(adc-480);
        degree = degree*90/240;
        //printf("degree:%5d\r",degree);
      }
      else{
        battery[0] = battery[1];
        battery[1] = battery[2];
        battery[2] = battery[3];
        battery[3] = battery[4];
        battery[4] = adc;
        vbatt = (battery[0]+battery[1]+battery[2]+battery[3]+battery[4])/5;
        //printf("Battery ADC: %d\n",vbatt);          // Print raw ADC value
        if(vbatt<LOW_BATT)
                printf("HL\n");
        accel = 1;
        ADMUX   &=      0xF0;                         // Reset ADC0 as input
      }
}


/* Timer1 Overflow Interrupt Vector */
/* 16-bit counter counts at 4Mhz/1024
 * When the counter overflows, the ADC input is changed
 * to the battery voltage, allowing us to read the current
 * voltage level roughly every 16 seconds.
 */
ISR(TIMER1_OVF_vect){
      accel = 0;
      ADMUX     |=      0x01;                 // Select ADC1 (VBAT) as input PA4
      start_adc();
}


/* Initialization Routine */
/* Initializes all necessary registers and enables
 * the necessary functionality.
 */
void initialize(void){
      stdout    =       stdin   =       &usart_str;   // Directs usart_str to stdout and stdin
      DDRA      =       0x02;                         // ADC Port inputs
      PORTA     =       0x00;
      UBRR0H    =    (unsigned char)(BAUD_VAL>>8);    // Set Baud Rate
      UBRR0L    =    (unsigned char)(BAUD_VAL&0x00FF);
      UCSR0B    =    (1<<RXEN0)|(1<<TXEN0);           // Enable RX & TX
      UCSR0B    |=   (1<<RXCIE0);                     // Enable RXC Interrupt
      UCSR0C    =    (3<<UCSZ00);                     // Set 8-N-1

      ADCSRA   |=  (1<<ADEN)|(1<<ADIE);               // Enable AD converter & interrupt
      ADCSRA   |=  (5<<ADPS0);                        // Set prescaler to 64
      ADCSRB    |=  (1<<ADLAR);                       // Left adjust result
      ADMUX    |=  (0<<REFS0);                        // Select VCC as reference
      ADMUX     &=      0xF0;                         // Select ADC0 (x) as input PA3

      TIMSK     |=      (1<<TOIE1);                   // Timer Overflow Interrupt Enable
      TCCR1B    |=      (5<<CS10);                    // Set Timer1 Clock Prescaler to 1024

      sei();                                          // Enable global interrupts
}

/* Start ADC Function */
/* Starts the ADC conversion then delays 1ms
 */
void start_adc(void){
      ADCSRA   |=  (1<<ADSC);                 // Start ADC conversion
      _delay_ms(1);                                   // Delay 1ms
}
```

```c
/* Decode Number Function */
/* Decodes ASCII character to actual number.
 * Returns value as an 8-bit binary number stored as an unsigned char.
 * In the case of 'R' and '\r' the function simply passes them through.
 */
unsigned char decode_number(unsigned char x){
      unsigned char temp=0;
      switch(x){
        case '0': temp=0; break;
        case '1': temp=1; break;
        case '2': temp=2; break;
        case '3': temp=3; break;
        case '4': temp=4; break;
        case '5': temp=5; break;
        case '6': temp=6; break;
        case '7': temp=7; break;
        case '8': temp=8; break;
        case '9': temp=9; break;
        case 'R': temp='R'; break;
        case '\r': temp='\r'; break;
        default: break;
      }
      return temp;
}
```

Figure 15.42: AVR source code for headset sub-system microcontroller.

```matlab
clear all
close all
clc

% delay=[10 1/20; 10 11; 20 22];
delay= [0 0];
load('Full_Sec_person_Avg.mat');
load('Half_Sec_person_Avg.mat');
load('Wall_Noise.mat');
load('Wall_Noise2.mat');
load('Value_Person.mat');
load('Car.mat');
load('Cane_Rotation_Test.mat');
[obj]=data_function(3, delay, 240, 360, 5, 20); %Import in "fake" data as strings

% Creates empty places to store Time, Type, and Values
% in individual columns.
t = zeros(length(obj),1);
type = cell(length(obj),1);
value = zeros(length(obj)/4,5);
% This for loop goes through each cell putting time, type, and value into
% the already created empty spaces.
k=1;
for i = 1:length(obj)
    temp = obj{i}; %Sets temp to the value of the cell at the i^th value of Data
    temp = temp(temp~=' '); %Looks through all of temp and gets rid of spaces
    ind1 = find(temp == 'H'); %Looks through the the temp to find if the type starts
with "H" or "G"
    ind2 = find(temp == 'G');
    if length(ind1) > length(ind2) %Since each i^th cell will only have "H" or "G"
ind1 or ind2 will have a value each time and not both
        ind = ind1;
    else
        ind = ind2;
    end
```

```
    value(k,1) = str2double(temp(1:ind-1)); %This convert string value of time to a
number.

    if strcmp(temp(ind:ind+1), 'HS')==1 %Look for where the data is from
        value(k,2)=str2double(temp(ind+2:end))/12; %Convert string after the heading
to number value
        if value(k,2)<1                              %place the values into values
matrix
            value(k,2)=0;
        end
    elseif strcmp(temp(ind:ind+1), 'HA')==1
        value(k,3)=str2double(temp(ind+2:end));
    elseif strcmp(temp(ind:ind+1), 'GS')==1
        value(k,4)=str2double(temp(ind+2:end))/12;
        if value(k,4)<1
            value(k,4)=0;
        end
    elseif strcmp(temp(ind:ind+1), 'GA')==1
        value(k,5)=str2double(temp(ind+2:end));
        k=k+1;
    end
end
```

Moving Average

```
ravg = zeros(length(value),2);
x=1;
% value=person;
% value=car;
while x<=length(value)
    if x<(length(value)-9)
        ravg(x,1) = mean(value(x:x+9,1));
        ravg(x,2) = mean(value(x:x+9,2));
        ravg(x,3) = mean(value(x:x+9,3));
        ravg(x,4) = mean(value(x:x+9,4));
        ravg(x,5) = mean(value(x:x+9,5));
        x=x+1;
    else
        ravg(x,1)= mean(value(x:end,1));
        ravg(x,2)= mean(value(x:end,2));
        ravg(x,3)= mean(value(x:end,3));
        ravg(x,4)= mean(value(x:end,4));
        ravg(x,5)= mean(value(x:end,5));
        x=x+1;end
end
x=2;
i=1;
j=1;
temp=6;
temp2=8;
temp3=30;
message=cell(1,3);
message2=cell(1,3);
% ravg=cane_test;
% ravg=f_sec_person_avg;
% ravg=h_sec_person_avg;
% ravg=wall_noise;
% ravg=wall_noise2;
while x<=length(ravg)
    if ravg(x,3)>=-20  && ravg(x,3)<=20  %Object must be within the visual parameters
```

```
        if ravg(x,2)>=0 && ravg(x,2)<=5 && ravg(x,2)<=temp-1 %Running avg. values are
checked to see if they are between 0-5ft
           message{i,1}=ravg(x,1);
           message{i,2}=ravg(x,2);
           message{i,3}=[num2str(round(message{i,1})) 's: ' 'Head level object is '
num2str(round(message{i,2})) 'ft ' 'ahead']; %Print to a messages file
           temp=message{i,2}; %Set to create the messages only when a foot apart
           i=i+1;
        elseif (ravg(x-1,2)>=0 && ravg(x-1,2)<=5) &&(ravg(x,2)-ravg(x-1,2)>0)
%Recognizes difference between objects
            message{i,1}=ravg(x,1);
            message{i,3}=[num2str(round(message{i,1})) 's: ' 'CLEAR'];
           i=i+1;
            temp=6;
        end
    end

    if ravg(x,5)>=-30 && ravg(x,5)<=30
        if ravg(x,4)>=0 && ravg(x,4)<=7 && ravg(x,4)<=temp2-1 %Running avg. values are
checked to see if they are between 0-7ft
            message2{j,1}=ravg(x,1);
            message2{j,2}=ravg(x,4);
            message2{j,3}=[num2str(round(message2{j,1})) 's: ' 'Ground level object is
' num2str(round(message2{j,2})) 'ft ' 'ahead'];
            temp2=message2{j,2};
            j=j+1;
        elseif (ravg(x-1,4)>=0 && ravg(x-1,4)<=7) && (ravg(x,4)-ravg(x-1,4)>0)
            message2{j,1}=ravg(x,1);
            message2{j,3}=[num2str(round(message2{j,1})) 's: ' 'CLEAR'];
           j=j+1;
                    temp2=8;
        end
    elseif (ravg(x,5)<=-30 || ravg(x,5)>=30) && ravg(x,4)<=temp3-1 %Situation if the
cane is not rotated correctly.
           if ravg(x,5)<=-30 && ravg(x,5)>=-180
               message2{j,1}=ravg(x,1);
               message2{j,2}=ravg(x,4);
               message2{j,3}=[num2str(round(message2{j,1})) 's: ' 'Rotate CW'];
               temp3=message2{j,2};
               j=j+1;
           elseif ravg(x,5)>=30 && ravg(x,5)<=180
              message2{j,1}=ravg(x,1);
              message2{j,2}=ravg(x,4);
              message2{j,3}=[num2str(round(message2{j,1})) 's: ' 'Rotate CCW'];
              temp3=message2{j,2};
              j=j+1;
           end
    end

x=x+1;
end
```

Figure 15.43: Preliminary data processing algorithm coded in MATLAB.

```
                case MESSAGE_READ://head:

                    char readBuf = (char)msg.arg1;
                    buffer[count]= readBuf;
                    count++;

                    if (readBuf=='\n')
                    {
                        buffer[count]='\0';
                        String readMessage= new String(buffer);
            //              mConversationArrayAdapter.add(readMessage);
                        count=0;
                        buffer= new char[buffer.length];
                        if (readMessage.startsWith(" HS")){
                           String buffer2= readMessage;
                            List<String> itemsInString =
Arrays.asList(buffer2.split(" "));

                    if (itemsInString.contains("HS")){
                        headList = (Double.parseDouble
(itemsInString.get(2))/12);

                    }
                    if (itemsInString.contains("HA")){
                        headAccelList =
(Double.parseDouble(itemsInString.get(4)));
                    }

                            //Sum of HS
                            sum_10 = sum_9;
                            sum_9 = sum_8;
                            sum_8 = sum_7;
                            sum_7 = sum_6;
                            sum_6 = sum_5;
                            sum_5 = sum_4;
                            sum_4 = sum_3;
                            sum_3 = sum_2;
                            sum_2 = sum_1;
                            sum_1 = headList;
                            double mean =
(sum_10+sum_9+sum_8+sum_7+sum_6+sum_5+sum_4+sum_3+sum_2+sum_1)/ 10;
                            runningAvg [0][0] =
Math.round(mean*1000.0)/1000.0;

                            //Sum of HA
                            sum2_10 = sum2_9;
                            sum2_9 = sum2_8;
                            sum2_8 = sum2_7;
                            sum2_7 = sum2_6;
                            sum2_6 = sum2_5;
                            sum2_5 = sum2_4;
                            sum2_4 = sum2_3;
                            sum2_3 = sum2_2;
                            sum2_2 = sum2_1;
                            sum2_1 = headAccelList;
```

```
                                double mean2 =
(sum2_10+sum2_9+sum2_8+sum2_7+sum2_6+sum2_5+sum2_4+sum2_3+sum2_2+sum2_1)/ 10;
                                runningAvg [0][1] =
Math.round(mean2*1000.0)/1000.0;

                        if(runningAvg[0][1]>=-20.0 && runningAvg[0][1]<=20.0){
                            if((runningAvg[0][0]>=0.0 && runningAvg[0][0]<=5.0)
&& runningAvg[0][0]<=(temp6-1.0)){
                                temp4 = runningAvg[0][0];
                                temp5 = Double.valueOf(Math.round(temp4*2));
                                message[0][0] = String.valueOf(((temp5)/2));
                                message[0][1] = ("Head level object is
"+(message[0][0])+"ft ahead");
                                if (temp4== 0.0){
                                    message[0][1] = null;
                                }
                                temp6 = temp4;
                            }
                            else if((runningAvg[0][0]-temp6)>0.0){
                                message[0][1] = ("CLEAR");
                                temp6=6.0;
                            }
                        }
                               if (message[0][1]== null){
                                   message[0][1]="nothing";
                               }
                                mConversationArrayAdapter.add(message[0][1]);

                            }
                        }
break;
```
Figure 15.44: Data processing algorithm initial Java implementation.

```
Changes to BluetoothChat code:
---------------------------------------------------------------------------
 char[] buffer = new char[100];
    int count=0;
---------------------------------------------------------------------------
case MESSAGE_READ:
                 char readBuf = (char)msg.arg1;
                         buffer[count]= readBuf;
                         count++;
                         if (readBuf=='\n')
                       {
                           String temp= new String(buffer);
                           mConversationArrayAdapter.add(temp);
                           count=0;
                           buffer= new char[buffer.length];
                          // if (temp.startsWith("HS")){}
                        }
---------------------------------------------------------------------------
private class ConnectedThread extends Thread {
        private final BluetoothSocket mmSocket;
        private final InputStream mmInStream;
        private final OutputStream mmOutStream;
        public ConnectedThread(BluetoothSocket socket) {
            Log.d(TAG, "create ConnectedThread");
            mmSocket = socket;
            InputStream tmpIn = null;
            OutputStream tmpOut = null;
            // Get the BluetoothSocket input and output streams
            try {
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e) {
                Log.e(TAG, "temp sockets not created", e);
            }
            mmInStream = tmpIn;
            mmOutStream = tmpOut;
        }
        public void run() {
            Log.i(TAG, "BEGIN mConnectedThread");
            byte[] buffer = new byte[1024];
            int bytes;
            // Keep listening to the InputStream while connected
            while (true) {
                try {
                    // Read from the InputStream
                    bytes = mmInStream.read();
//buffer
                    // Send the obtained bytes to the UI Activity
         mHandler.obtainMessage(BluetoothChat.MESSAGE_READ, bytes, -1, bytes)
                            .sendToTarget();
                } catch (IOException e) {
                    Log.e(TAG, "disconnected", e);
                    connectionLost();
                    break;
                }}
```

Figure 15.45: Changes made to Bluetooth Chat example.

Appendix I: Datasheets

Atmel

8-bit Atmel tinyAVR Microcontroller with 16K Bytes In-System Programmable Flash

ATtiny1634

Features

- High Performance, Low Power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 125 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
- High Endurance, Non-volatile Memory Segments
  - 16K Bytes of In-System, Self-Programmable Flash Program Memory
    - Endurance: 10,000 Write/Erase Cycles
  - 256 Bytes of In-System Programmable EEPROM
    - Endurance: 100,000 Write/Erase Cycles
  - 1K Byte of Internal SRAM
  - Data retention: 20 years at 85°C / 100 years at 25°C
  - Programming Lock for Self-Programming Flash & EEPROM Data Security
- Peripheral Features
  - Dedicated Hardware and QTouch® Library Support for Capacitive Touch Sensing
  - One 8-bit and One 16-bit Timer/Counter with Two PWM Channels, Each
  - 12-channel, 10-bit ADC
  - Programmable Ultra Low Power Watchdog Timer
  - On-chip Analog Comparator
  - Two Full Duplex USARTs with Start Frame Detection
  - Universal Serial Interface
  - Slave I^2C Serial Interface
- Special Microcontroller Features
  - debugWIRE On-chip Debug System
  - In-System Programmable via SPI Port
  - Internal and External Interrupt Sources
    - Pin Change Interrupt on 18 Pins
  - Low Power Idle, ADC Noise Reduction, Standby and Power-down Modes
  - Enhanced Power-on Reset Circuit
  - Programmable Brown-out Detection Circuit with Supply Voltage Sampling
  - Calibrated 8MHz Oscillator with Temperature Calibration Option
  - Calibrated 32kHz Ultra Low Power Oscillator
  - On-chip Temperature Sensor
- I/O and Packages
  - 18 Programmable I/O Lines
  - 20-pad QFN/MLF, and 20-pin SOIC
- Operating Voltage:
  - 1.8 – 5.5V
- Speed Grade:
  - 0 – 2MHz @ 1.8 – 5.5V
  - 0 – 8MHz @ 2.7 – 5.5V
  - 0 – 12MHz @ 4.5 – 5.5V
- Temperature Range: -40°C to +85°C
- Low Power Consumption
  - Active Mode: 0.2mA at 1.8V and 1MHz
  - Idle Mode: 30µA at 1.8V and 1MHz
  - Power-Down Mode (WDT Enabled): 1µA at 1.8V
  - Power-Down Mode (WDT Disabled): 100nA at 1.8V

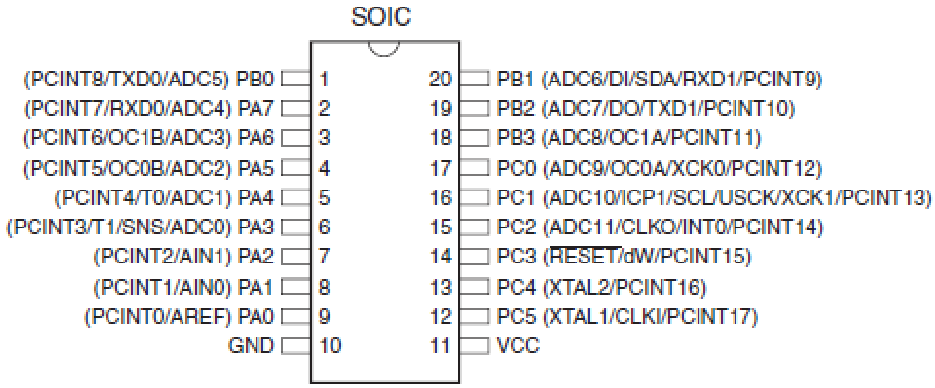
8303F–AVR–08/2013

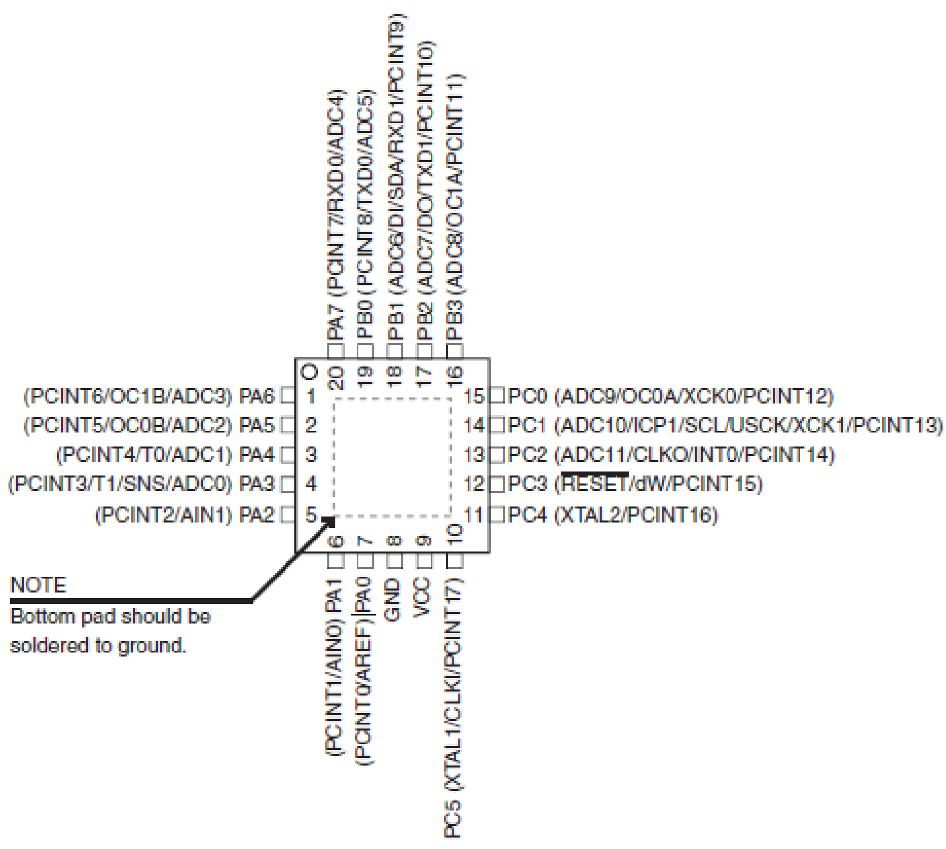
Datasheet 1: ATtiny1634 Datasheet

http://www.atmel.com/Images/Atmel-8303-8-bit-AVR-Microcontroller-tinyAVR-ATtiny1634_Datasheet.pdf

1. Pin Configurations

Figure 1-1. Pinout of ATtiny1634





063450

LI-POLYMER BATTERY PACKS

Specification

Type:063450 1000mAh

Prepared/Date	Auditing/Date	Approved/Date
WANG MAR 16, 2006	LI MAR 16, 2006	XIONG MAR 16, 2006

Datasheet 2: 3.7V 1000mAh Li-Po Battery Datasheet

https://www.sparkfun.com/datasheets/Batteries/UnionBattery-1000mAh.pdf

UNIONFORTUNE PRODUCT SPECIFICATION	Doc. No.	2006.3.16
	Edition No.	2.0
	Sheet	1/5

1□Scope

This product specification describes UNIONFORTUNE polymer lithium-ion battery. Please using the test methods that recommend in this specification. If you have any opinions or advices about the test items and methods, please contact us. Please read the cautions recommended in the specifications first, take the credibility measure of the cell's using.

2□Product Type, Model and Dimension

2.1 Type□Polymer lithium-ion battery

2.2 Model□063450

2.3 Cell Dimension(Max, Thickness×Width×Length□mm^3)□ 6×34×50

Pack Dimension(Max, Thickness×Width×Length□mm^3)□ None

3□Specification

Item		Specifications	Remark
Nominal Capacity		1000 mAh	0.2C_5A discharge
Nominal Voltage		3.7V	Average Voltage at 0.2C_5A discharge
Charge Current		Standard□0.2 C_5A□Max□1C_5A	Working temperature□0□40□
Charge cut-off Voltage		4.20±0.03V	
Standard Discharge Current		0.2C_5A	Working temperature□-20□60□
Max Discharge Current		2.0C_5A	Working temperature□0□60□
Discharge cut-off Voltage		2.75 V	
Cell Voltage		3.7-3.9 V	When leave factory
Impedance		≤ 300 mΩ	AC 1KHz after 50% charge
Weight		Approx: 20g	
Storage temperature	≤1month	-20□45□	Best 20±5□ for long-time storage
	≤3month	0□30□	
	≤6month	20±5□	
Storage humidity		65±20% RH	

4□General Performance

Definition of Standard charging method□At 20±5□□charging the cell initially with constant current 0.2C_5A till voltage 4.2V, then with constant voltage 4.2V till current declines to 0.05C_5A.

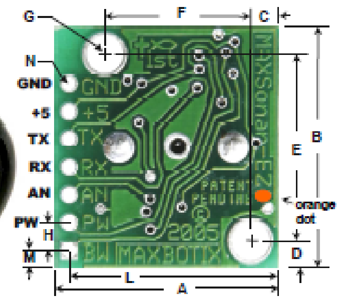
Item		Test Methods	Performance
4.1	0.2C Capacity	After standard charging, laying the battery 0.5h, then discharging at 0.2C_5A to voltage 2.75V, recording the discharging time.	≥300min
4.2	1C Capacity	After standard charging, laying the battery 0.5h, then discharging at 1C_5A to voltage 2.75V, recording the discharging time.	≥51min
4.3	Cycle Life	Constant current 1C_5A charge to 4.2V, then constant voltage charge to current declines to 0.05C_5A, stay 5min □constant current 1C_5A discharge to 2.75V □stay 5min. Repeat above steps till continuously discharging time less than 36min.	≥300times
4.4	Capability of keeping electricity	20±5□ After standard charging, laying the battery 28days, discharging at 0.2C_5A to voltage 2.75V, recording the discharging time.	≥240min

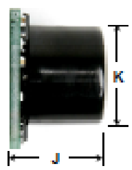
MB1030

LV-MaxSonar®-EZ3™ High Performance Sonar Range Finder

RoHS

With 2.5V - 5.5V power, the LV-MaxSonar®-EZ3™ provides very short to long-range detection and ranging, in an incredibly small package. The LV-MaxSonar®-EZ3™ detects objects from 0-inches to 254-inches (6.45-meters) and provides sonar range information from 6-inches out to 254-inches with 1-inch resolution. Objects from 0-inches to 6-inches range as 6-inches. The interface output formats included are pulse width output, analog voltage output, and serial digital output.





approximately actual size

A	0.785"	19.9 mm		H	0.100"	2.54 mm
B	0.870"	22.1 mm		J	0.610"	15.5 mm
C	0.100"	2.54 mm		K	0.645"	16.4 mm
D	0.100"	2.54 mm		L	0.735"	18.7 mm
E	0.670"	17.0 mm		M	0.065"	1.7 mm
F	0.510"	12.6 mm		N	0.038" dia.	1.0 mm dia.
G	0.124" dia.	3.1 mm dia.		weight, 4.3 grams		

values are nominal

Features
- Continuously variable gain for beam control and side lobe suppression
- Object detection includes zero range objects
- 2.5V to 5.5V supply with 2mA typical current draw
- Readings can occur up to every 50mS, (20-Hz rate)
- Free run operation can continually measure and output range information
- Triggered operation provides the range reading as desired
- All interfaces are active simultaneously
- Serial, 0 to Vcc, 9600Baud, 81N
- Analog, (Vcc/512) / inch
- Pulse width, (147uS/inch)
- Learns ringdown pattern when commanded to start ranging
- Designed for protected indoor environments
- Sensor operates at 42KHz
- High output square wave sensor drive (double Vcc)

Benefits
- Very low cost sonar ranger
- Reliable and stable range data
- Sensor dead zone virtually gone
- Lowest power ranger
- Quality beam characteristics
- Mounting holes provided on the circuit board
- Very low power ranger, excellent for multiple sensor or battery based systems
- Can be triggered externally or internally
- Sensor reports the range reading directly, frees up user processor
- Fast measurement cycle
- User can choose any of the three sensor outputs

Beam Characteristics
Many applications require a narrower beam or lower sensitivity than the LV-MaxSonar®-EZ1™. MaxBotix® Inc., is offering, the EZ2™, EZ3™, & EZ4™ with progressively narrower beam angles allowing the sensor to match the application. Sample results for the LV-MaxSonar®-EZ3™ measured beam patterns are shown below on a 12-inch grid. The detection pattern is shown for;

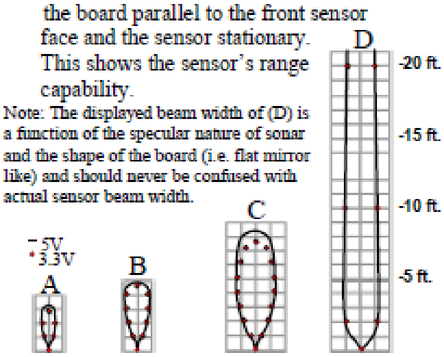
(A) 0.25-inch diameter dowel, note the narrow beam for close small objects,

(B) 1-inch diameter dowel, note the long narrow detection pattern,

(C) 3.25-inch diameter rod, note the long controlled detection pattern,

(D) 11-inch wide board moved left to right with the board parallel to the front sensor face and the sensor stationary. This shows the sensor's range capability.

Note: The displayed beam width of (D) is a function of the specular nature of sonar and the shape of the board (i.e. flat mirror like) and should never be confused with actual sensor beam width.



beam characteristics are approximate

MaxBotix® Inc.

MaxBotix, MaxSonar, EZ2, EZ3 & EZ4 are trademarks of MaxBotix Inc.

Page 1

Email: info@maxbotix.com
Web: www.maxbotix.com

PD10007e

Datasheet 3: Ultrasonic Sensor Datasheet

http://www.maxbotix.com/documents/MB1030_Datasheet.pdf

LV-MaxSonar®-EZ3™ Pin Out

GND – Return for the DC power supply. GND (& Vcc) must be ripple and noise free for best operation.

+5V – Vcc – Operates on 2.5V - 5.5V. Recommended current capability of 3mA for 5V, and 2mA for 3V.

TX – When the *BW is open or held low, the TX output delivers asynchronous serial with an RS232 format, except voltages are 0-Vcc. The output is an ASCII capital "R", followed by three ASCII character digits representing the range in inches up to a maximum of 255, followed by a carriage return (ASCII 13). The baud rate is 9600, 8 bits, no parity, with one stop bit. Although the voltage of 0-Vcc is outside the RS232 standard, most RS232 devices have sufficient margin to read 0-Vcc serial data. If standard voltage level RS232 is desired, invert, and connect an RS232 converter such as a MAX232. When BW pin is held high the TX output sends a single pulse, suitable for low noise chaining. (no serial data).

RX – This pin is internally pulled high. The EZ3™ will continually measure range and output if RX data is left unconnected or held high. If held low the EZ3™ will stop ranging. Bring high for 20uS or more to command a range reading.

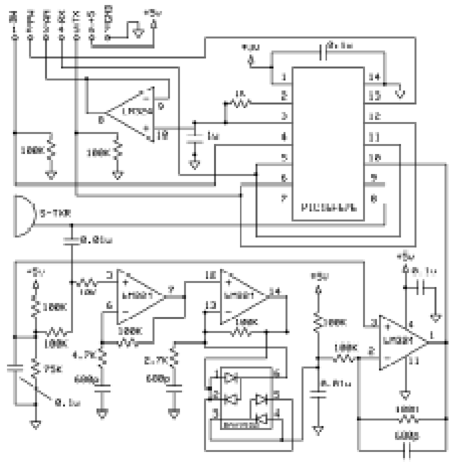
AN – Outputs analog voltage with a scaling factor of (Vcc/512) per inch. A supply of 5V yields ~9.8mV/in. and 3.3V yields ~6.4mV/in. The output is buffered and corresponds to the most recent range data.

PW – This pin outputs a pulse width representation of range. The distance can be calculated using the scale factor of 147uS per inch.

BW – *Leave open or hold low for serial output on the TX output. When BW pin is held high the TX output sends a pulse (instead of serial data), suitable for low noise chaining.

LV-MaxSonar®-EZ3™ Circuit

The LV-MaxSonar®-EZ3™ sensor functions using active components consisting of an LM324, a diode array, a PIC16F676, together with a variety of passive components.



LV-MaxSonar®-EZ3™ Timing Description

250mS after power-up, the LV-MaxSonar®-EZ3™ is ready to accept the RX command. If the RX pin is left open or held high, the sensor will first run a calibration cycle (49mS), and then it will take a range reading (49mS). After the power up delay, the first reading will take an additional ~100mS. Subsequent readings will take 49mS. The LV-MaxSonar®-EZ3™ checks the RX pin at the end of every cycle. Range data can be acquired once every 49mS.

Each 49mS period starts by the RX being high or open, after which the LV-MaxSonar®-EZ3™ sends thirteen 42KHz waves, after which the pulse width pin (PW) is set high. When a target is detected the PW pin is pulled low. The PW pin is high for up to 37.5mS if no target is detected. The remainder of the 49mS time (less 4.7mS) is spent adjusting the analog voltage to the correct level. When a long distance is measured immediately after a short distance reading, the analog voltage may not reach the exact level within one read cycle. During the last 4.7mS, the serial data is sent. The LV-MaxSonar®-EZ3™ timing is factory calibrated to one percent at five volts, and in use is better than two percent. In addition, operation at 3.3V typically causes the objects range, to be reported, one to two percent further than actual.

LV-MaxSonar®-EZ3™ General Power-Up Instruction

Each time after the LV-MaxSonar®-EZ3™ is powered up, it will calibrate during its first read cycle. The sensor uses this stored information to range a close object. It is important that objects not be close to the sensor during this calibration cycle. The best sensitivity is obtained when it is clear for fourteen inches, but good results are common when clear for at least seven inches. If an object is too close during the calibration cycle, the sensor may then ignore objects at that distance.

The LV-MaxSonar®-EZ3™ does not use the calibration data to temperature compensate for range, but instead to compensate for the sensor ringdown pattern. If the temperature, humidity, or applied voltage changes during operation, the sensor may require recalibration to reacquire the ringdown pattern. Unless recalibrated, if the temperature increases, the sensor is more likely to have false close readings. If the temperature decreases, the sensor is more likely to have reduced up close sensitivity. To recalibrate the LV-MaxSonar®-EZ3™, cycle power, then command a read cycle.

Email: info@maxbotix.com
Web: www.maxbotix.com

Resistance Weld Thru-Hole Crystal

FOX

Model: HC49SLF | **RoHS Compliant / Pb Free** | **Rev. 5/2/2013**

Page 1 of 2

http://www.foxonline.com/need_a_sample.htm

Need a Sample®

FEATURES
- Low Cost
- Stocking Standard (see page 2)
- Fundamental to 40 MHz
- Resistance Weld

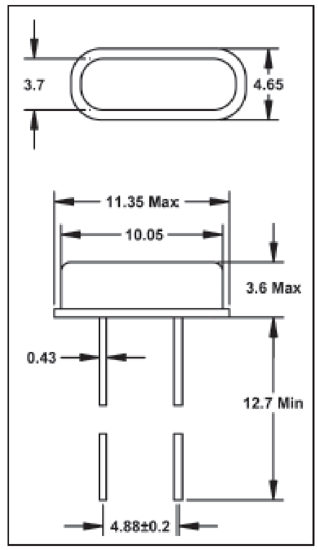
OPTIONS
- Tolerances to 10 PPM
- Stabilities to 5 PPM
- Temperatures to -55°C ~ +125°C
- 2.5mm Height Max (HC49SSLF)

• PART NUMBER Learn More - Internet Required

Part Number	Model Number	Frequency Range (MHz)
277LF-Frequency-xxxxx	HC49SLF	3.200 ~ 80.000

• STANDARD SPECIFICATIONS[1]

PARAMETERS	MAX (unless otherwise noted)
Frequency Range	3.200 ~ 80.000 MHz
Frequency Tolerance @ 25°C	±50 PPM [1,3]
Frequency Stability, ref @ 25°C	±50 PPM [1]
Temperature Range Operating (TOPR) Storage (TSTG)	 -20°C ~ +70°C [2] -40°C ~ +125°C
Shunt Capacitance (Co)	7pF
Load Capacitance (CL)	10 pF ~ Series (Customer Specified)
Drive Level	100uW
Aging per year	±3 PPM
Maximum Soldering Temp / Time	260°C / 10 Seconds
Moisture Sensitivity Level (MSL)	1
Termination Finish:	Sn/Ag/Cu



All dimensions are in millimeters.
TERMINATION FINISH: Sn/3.0Ag/0.5Cu

[1] Other tolerances, stabilities available. Consult Fox Customer Service for specific requirements.
[2] Operating temperature ranges to -55°C ~ +125°C available.
[3] Crystal has a ±30ppm tolerance specification for all standard part numbers. These part numbers are listed on page 2 of this datasheet.

Note: Dimensional drawing is for reference to critical specifications defined by size measure¬ments. Certain non-critical visual attributes, such as side castellations, etc. may vary.
All specifications subject to change without notice.

Frequency Range (MHz)	Operating Mode	Max ESR Ω
3.200 ~ 3.500	Fundamental	300
3.500+ ~ 4.000	Fundamental	200
4.000+ ~ 5.000	Fundamental	150
5.000+ ~ 6.000	Fundamental	120
6.000+ ~ 7.000	Fundamental	100
7.000+ ~ 9.000	Fundamental	80
9.000+ ~ 13.000	Fundamental	60
13.000+ ~ 20.000	Fundamental	40
20.000+ ~ 40.000	Fundamental	30
27.000 ~ 70.000	3rd OT	100
70.000+ ~ 80.000	3rd OT	80

FOXElectronics 5570 Enterprise Parkway Fort Myers, Florida 33905 USA +1.239.693.0099 FAX +1.239.693.1554 www.foxonline.com
EMEA Tel: +44 .1283.568153 | Canada +1.888.438.2369 | Asia Hong Kong Tel: +852.2854.4285 | Japan Tel: +81.3.3374.2079

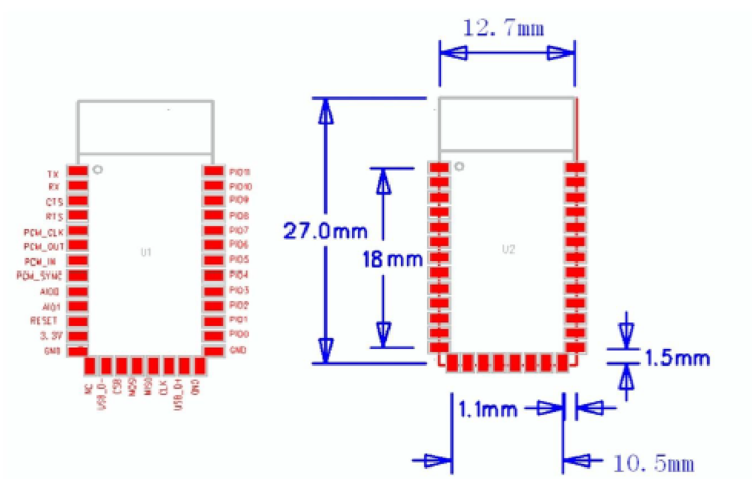
Datasheet 4: 4MHz Crystal Oscillator Datasheet

http://www.mouser.com/Search/ProductDetail.aspx?R=FOXSLF/040virtualkey55910000virtualkey559-FOXS040-LF

Software features

- Default Baud rate: 38400, Data bits:8, Stop bit:1,Parity:No parity, Data control: has. Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.
- Given a rising pulse in PIO0, device will be disconnected.
- Status instruction port PIO1: low-disconnected, high-connected;
- PIO10 and PIO11 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:"0000" as default
- Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

Hardware



Datasheet 5: HC-05 Bluetooth Transceiver Datasheet

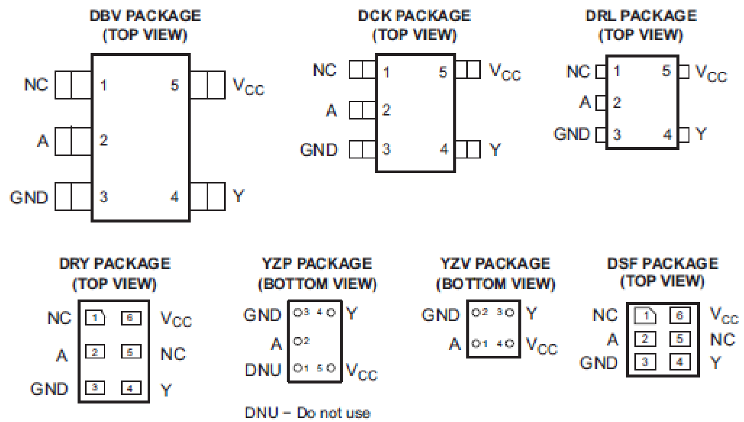
ftp://imall.iteadstudio.com/Modules/IM120723009/DS_IM120723009.pdf

SINGLE SCHMITT-TRIGGER INVERTER

Check for Samples: SN74LVC1G14

FEATURES

- Available in the Texas Instruments NanoStar™ and NanoFree™ Packages
- Supports 5-V V_{CC} Operation
- Inputs Accept Voltages to 5.5 V
- Max t_{pd} of 4.6 ns at 3.3 V
- Low Power Consumption, 10-μA Max I_{CC}
- ±24-mA Output Drive at 3.3 V
- I_{off} Supports Live Insertion, Partial Power Down Mode, and Back Drive Protection
- Latch-Up Performance Exceeds 100 mA Per JESD 78, Class II
- ESD Protection Exceeds JESD 22
  - – 2000-V Human-Body Model (A114-A)
  - – 200-V Machine Model (A115-A)
  - – 1000-V Charged-Device Model (C101)



NC – No internal connection

See mechanical drawings for dimensions.

DESCRIPTION/ORDERING INFORMATION

This single Schmitt-trigger inverter is designed for 1.65-V to 5.5-V V_{CC} operation.

The SN74LVC1G14 device contains one inverter and performs the Boolean function $Y = \overline{A}$. The device functions as an independent inverter, but because of Schmitt action, it may have different input threshold levels for positive-going (V_{T+}) and negative-going (V_{T-}) signals.

NanoStar™ and NanoFree™ package technology is a major breakthrough in IC packaging concepts, using the die as the package.

This device is fully specified for partial-power-down applications using I_{off}. The I_{off} circuitry disables the outputs, preventing damaging current backflow through the device when it is powered down.

Datasheet 6: Inverter Datasheet

http://www.ti.com/lit/ds/sces647a/sces647a.pdf

Absolute Maximum Ratings[1]

over operating free-air temperature range (unless otherwise noted)

			MIN	MAX	UNIT
V_{CC}	Supply voltage range		–0.5	6.5	V
V_I	Input voltage range[2]		–0.5	6.5	V
V_O	Voltage range applied to any output in the high-impedance or power-off state[2]		–0.5	6.5	V
V_O	Voltage range applied to any output in the high or low state[2] [3]		–0.5	V_{CC} + 0.5	V
I_{IK}	Input clamp current	$V_I < 0$		–50	mA
I_{OK}	Output clamp current	$V_O < 0$		–50	mA
I_O	Continuous output current			±50	mA
	Continuous current through V_{CC} or GND			±100	mA
θ_{JA}	Package thermal impedance[4]	DBV package		206	°C/W
		DCK package		252	
		DRL package		142	
		DRY package		234	
		DSF package		300	
		YZP package		132	
		YZV package		123	
T_{stg}	Storage temperature range		–65	150	°C

(1) Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) The input and output negative-voltage ratings may be exceeded if the input and output current ratings are observed.
(3) The value of V_{CC} is provided in the recommended operating conditions table.
(4) The package thermal impedance is calculated in accordance with JESD 51-7.

Recommended Operating Conditions[1]

			MIN	MAX	UNIT
V_{CC}	Supply voltage	Operating	1.65	5.5	V
		Data retention only	1.5		
V_I	Input voltage		0	5.5	V
V_O	Output voltage		0	V_{CC}	V
I_{OH}	High-level output current	V_{CC} = 1.65 V		–4	mA
		V_{CC} = 2.3 V		–8	
		V_{CC} = 3 V		–16	
				–24	
		V_{CC} = 4.5 V		–32	
I_{OL}	Low-level output current	V_{CC} = 1.65 V		4	mA
		V_{CC} = 2.3 V		8	
		V_{CC} = 3 V		16	
				24	
		V_{CC} = 4.5 V		32	
T_A	Operating free-air temperature		–40	85	°C

(1) All unused inputs of the device must be held at V_{CC} or GND to ensure proper device operation. Refer to the TI application report, *Implications of Slow or Floating CMOS Inputs*, literature number SCBA004.

Freescale Semiconductor
Technical Data

Document Number: MMA7361L
Rev 0, 04/2008

RoHS

±1.5g, ±6g Three Axis Low-g Micromachined Accelerometer

MMA7361L

MMA7361L: XYZ AXIS ACCELEROMETER ±1.5g, ±6g

The MMA7361L is a low power, low profile capacitive micromachined accelerometer featuring signal conditioning, a 1-pole low pass filter, temperature compensation, self test, 0g-Detect which detects linear freefall, and g-Select which allows for the selection between 2 sensitivities. Zero-g offset and sensitivity are factory set and require no external devices. The MMA7361L includes a Sleep Mode that makes it ideal for handheld battery powered electronics.

Features

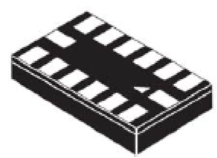
- 3mm x 5mm x 1.0mm LGA-14 Package
- Low Current Consumption: 400 µA
- Sleep Mode: 3 µA
- Low Voltage Operation: 2.2 V – 3.6 V
- High Sensitivity (800 mV/g @ 1.5g)
- Selectable Sensitivity (±1.5g, ±6g)
- Fast Turn On Time (0.5 ms Enable Response Time)
- Self Test for Freefall Detect Diagnosis
- 0g-Detect for Freefall Protection
- Signal Conditioning with Low Pass Filter
- Robust Design, High Shocks Survivability
- RoHS Compliant
- Environmentally Preferred Product
- Low Cost

Typical Applications

- 3D Gaming: Tilt and Motion Sensing, Event Recorder
- HDD MP3 Player: Freefall Detection
- Laptop PC: Freefall Detection, Anti-Theft
- Cell Phone: Image Stability, Text Scroll, Motion Dialing, E-Compass
- Pedometer: Motion Sensing
- PDA: Text Scroll
- Navigation and Dead Reckoning: E-Compass Tilt Compensation
- Robotics: Motion Sensing

ORDERING INFORMATION				
Part Number	**Temperature Range**	**Package Drawing**	**Package**	**Shipping**
MMA7361LT	–40 to +85°C	1977-01	LGA-14	Tray
MMA7361LR1	–40 to +85°C	1977-01	LGA-14	7" Tape & Reel
MMA7361LR2	–40 to +85°C	1977-01	LGA-14	13" Tape & Reel

Bottom View



14 LEAD LGA CASE 1977-01

Top View

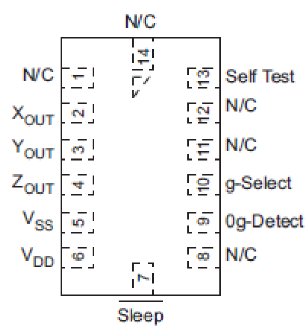


Figure 1. Pin Connections

Datasheet 7: MMA7361 Accelerometer Datasheet

https://www.sparkfun.com/datasheets/Components/General/MMA7361L.pdf

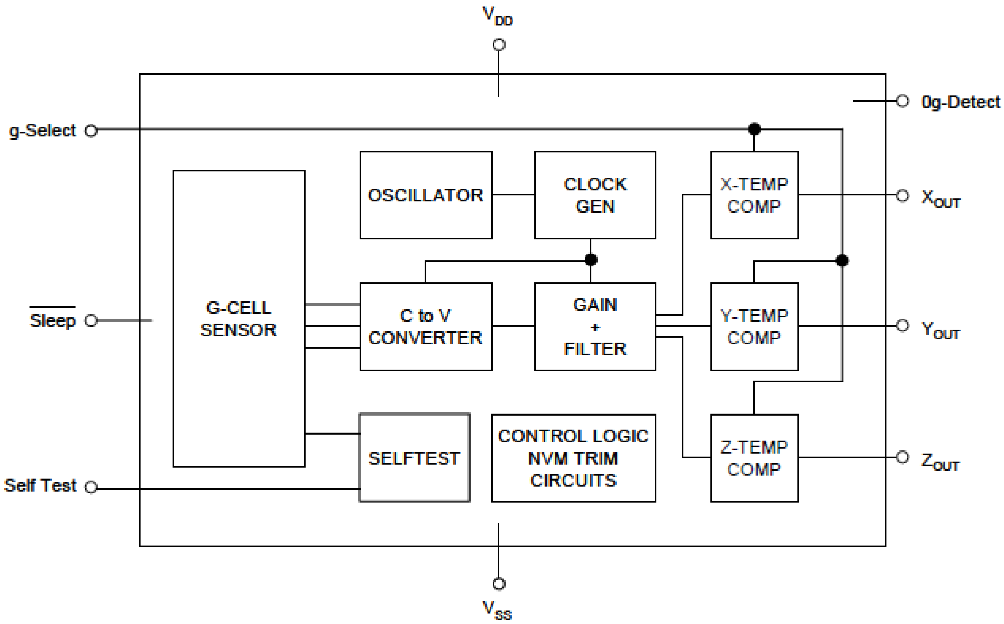


Figure 2. Simplified Accelerometer Functional Block Diagram

Table 1. Maximum Ratings

(Maximum ratings are the limits to which the device can be exposed without causing permanent damage.)

Rating	Symbol	Value	Unit
Maximum Acceleration (all axis)	g_{max}	±5000	g
Supply Voltage	V_{DD}	–0.3 to +3.6	V
Drop Test[1]	D_{drop}	1.8	m
Storage Temperature Range	T_{stg}	–40 to +125	°C

1. Dropped onto concrete surface from any axis.

ELECTRO STATIC DISCHARGE (ESD)

WARNING: This device is sensitive to electrostatic discharge.

Although the Freescale accelerometer contains internal 2000 V ESD protection circuitry, extra precaution must be taken by the user to protect the chip from ESD. A charge of over 2000 volts can accumulate on the human body or associated test equipment. A charge of this magnitude can alter the performance or cause failure of the chip. When handling the accelerometer, proper ESD precautions should be followed to avoid exposing the device to discharges which may be detrimental to its performance.

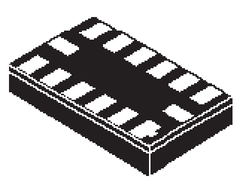
Soldering and Mounting Guidelines for the LGA Accelerometer Sensor to a PC Board

by: Kimberly Tuck, Cheol Han
Sensors and Actuator Solutions Division
Tempe, AZ

INTRODUCTION

MEMS based sensors are sensitive to Printed Circuit Board (PCB) reflow processes. For optimal zero-g offset after PCB mounting, care must be taken to PCB layout and reflow conditions. This application note is a guideline for soldering and mounting the LGA package inertial sensors. The purpose of these guidelines is to minimize the stress on the package after board mounting. Both the MMA73x1L 3-axis analog output family of accelerometers and the MMA745xL digital output accelerometer use the Land Grid Array (LGA) package platform. This application note describes suggested methods of soldering these devices to the PC board for consumer applications. Figure 1 shows the bottom view of LGA 14 lead, 3 x 5 mm individual sensor device. Figure 2 shows the recommended PCB land pattern for the package.

Bottom View



14 LEAD
LGA
CASE 1935-01

Figure 1. LGA 14-Lead, 5 x 3 mm Die Sensor

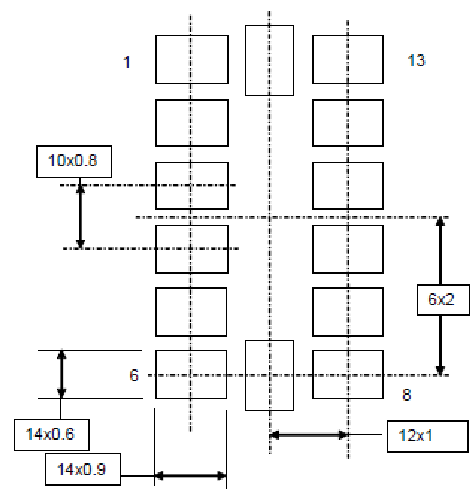


Figure 2. Recommended PCB Land Pattern for the 5 x 3 mm LGA Package

OVERVIEW OF SOLDERING CONSIDERATIONS

Information provided here is based on experiments executed on LGA devices. They do not represent exact conditions present at a customer site. Hence, information herein should be used as a guidance only and process and design optimizations are recommended to develop an application specific solution. It should be noted that with the proper PCB footprint and solder stencil designs the package will self-align during the solder reflow process.

The following are the recommended guidelines to follow for mounting LGA sensors for consumer applications.

Datasheet 8: SMD Accelerometer Soldering Guide

http://www.freescale.com/files/sensors/doc/app_note/AN3484.pdf

PCB Mounting Recommendations

1. The PCB land should be designed with Non Solder Mask Defined (NSMD) as shown in Figure 5.
2. No additional metal pattern underneath package as shown in Figure 4.

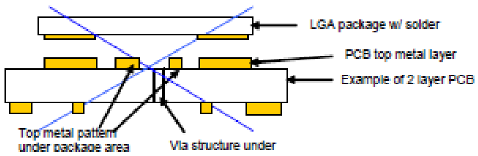


Figure 3. Incorrect PCB Top Metal Pattern Under Package

3. PCB land pad is 0.9mm x 0.6mm which is the size of the package pad plus 0.1mm as shown below in Figure 5.

Figure 4. Correct PCB Top Metal Pattern Under Package

4. The solder mask opening is equal to the size of the PCB land pad plus an extra 0.1mm as shown in Figure 5.
5. The stencil aperture size is equal to the PCB land pad – 0.025mm. Also note that for the 4 corner pads the aperture size must be larger for solder balancing as shown in Figure 6 and Figure 7. A 6mil thick stencil is recommended.

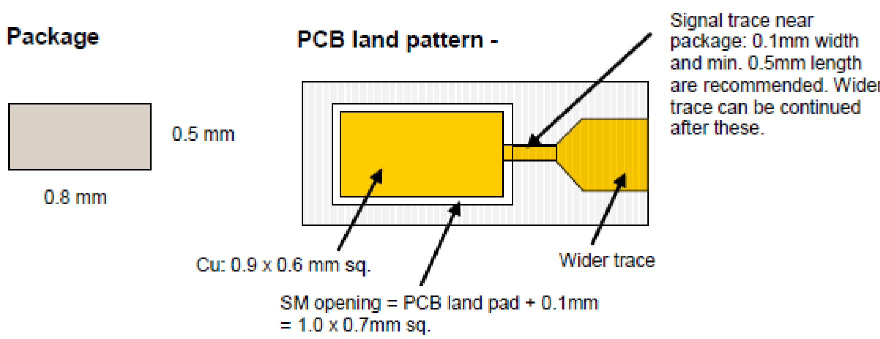


Figure 5. Recommended PCB Land Pad, Solder Mask, and Signal Trace Near Package Design

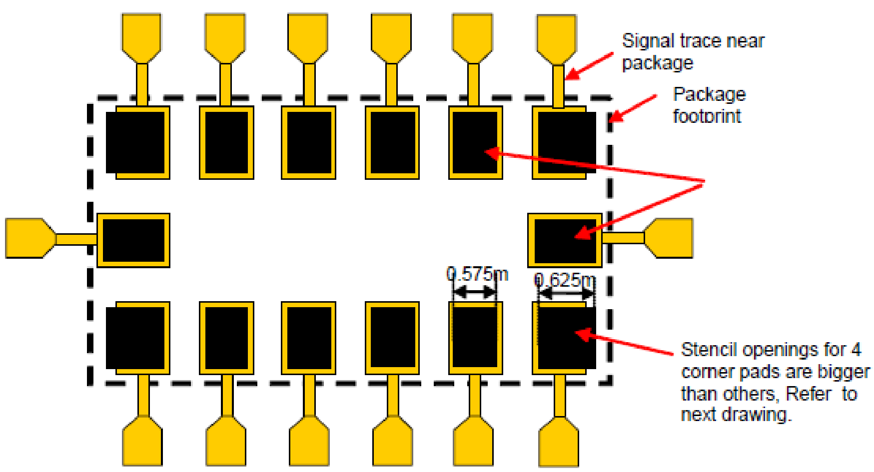


Figure 6. Stencil Design Guidelines

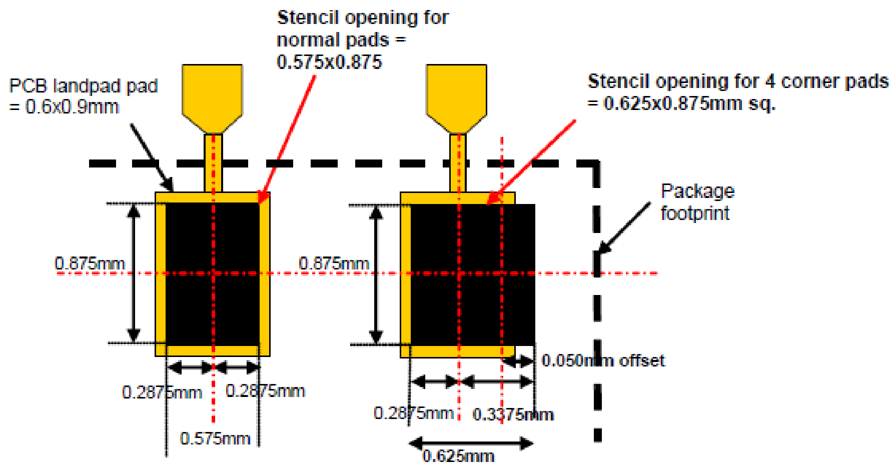


Figure 7. Stencil Design Guidelines (detailed dimensions for corner pads)

6. Do not place any components or vias at a distance less than 2mm from the package land area. This may cause additional package stress if it is too close to the package land area.
7. Signal traces connected to pads should be as symmetric as possible. Put dummy traces on N/C pads in order to have same length of exposed trace for all pads. Signal traces with 0.1mm width and min. 0.5mm length for all PCB land pads near the package are recommended as shown in Figure 5, Figure 6, and Figure 7. Wider trace can be continued after the 0.5mm zone.
8. Use a standard pick and place process and equipment. Do not us a hand soldering process.
9. It is recommended to use a cleanable solder paste with an additional cleaning step after SMT mount.
10. Do not use a screw down or stacking to fix the PCB into an enclosure because this could bend the PCB putting stress on the package.
11. The PCB should be rated for the multiple lead-free reflow condition with max 260°C temperature.
12. The recommended peak temperature for the solder paste for lead free (Pb-free) is 245°C - 250°C and for the tin-lead (Sn-Pb), 215°C - 225°C.

Please cross reference with the device data sheet for mounting guidelines specific to the exact device used.

Freescale LGA sensors are compliant with Restrictions on Hazardous Substances (RoHS), having halide free molding compound (green) and lead-free terminations. These terminations are compatible with tin-lead (Sn-Pb) as well as tin-silver-copper (Sn-Ag-Cu) solder paste soldering processes. Reflow profiles applicable to those processes can be used successfully for soldering the devices.

SUMMARY

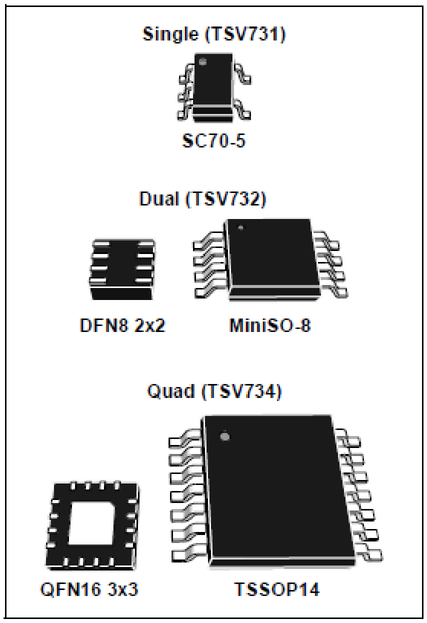
There are many new applications being designed using LGA die accelerometers. This document suggests soldering methods for the MMA73x1L family and the MMA745xL accelerometers.

TSV731, TSV732, TSV734

life.augmented

High accuracy (200 µV) micropower 60 µA, 900 kHz 5 V CMOS operational amplifiers

Datasheet - preliminary data



Features

- Low offset voltage: 200 µV max.
- Low power consumption: 60 µA at 5 V
- Low supply voltage: 1.5 V to 5.5 V
- Gain bandwidth product: 900 kHz typ.
- Low input bias current: 1 pA typ.
- Rail-to-rail input and output
- EMI hardened operational amplifiers
- High tolerance to ESD: 4 kV HBM
- Extended temperature range: -40 to +125 °C

Benefits

- Higher accuracy without calibration
- Energy saving
- Guaranteed operation on low-voltage battery

Related products

- See the TSV71 series (150 kHz for 14 µA) for more power savings

Applications

- Battery powered applications
- Portable devices
- Signal conditioning
- Active filtering
- Medical instrumentation

Description

The TSV73x series of single, dual, and quad operational amplifiers offer low-voltage operation, rail-to-rail input and output, and excellent accuracy (V_{io} lower than 200 µV at 25 °C).

These devices benefit from STMicroelectronics® 5 V CMOS technology and offer an excellent speed/power consumption ratio (900 kHz typical gain bandwidth) while consuming 60 µA typical at 5 V. The TSV73x series also feature an ultra-low input bias current.

The single version (TSV731), the dual version (TSV732), and the quad version (TSV734) are housed in the smallest industrial packages.

These characteristics make the TSV73x family ideal for sensor interfaces, battery-powered and portable applications, and active filtering.

March 2013 | DocID023708 Rev 2 | 1/29

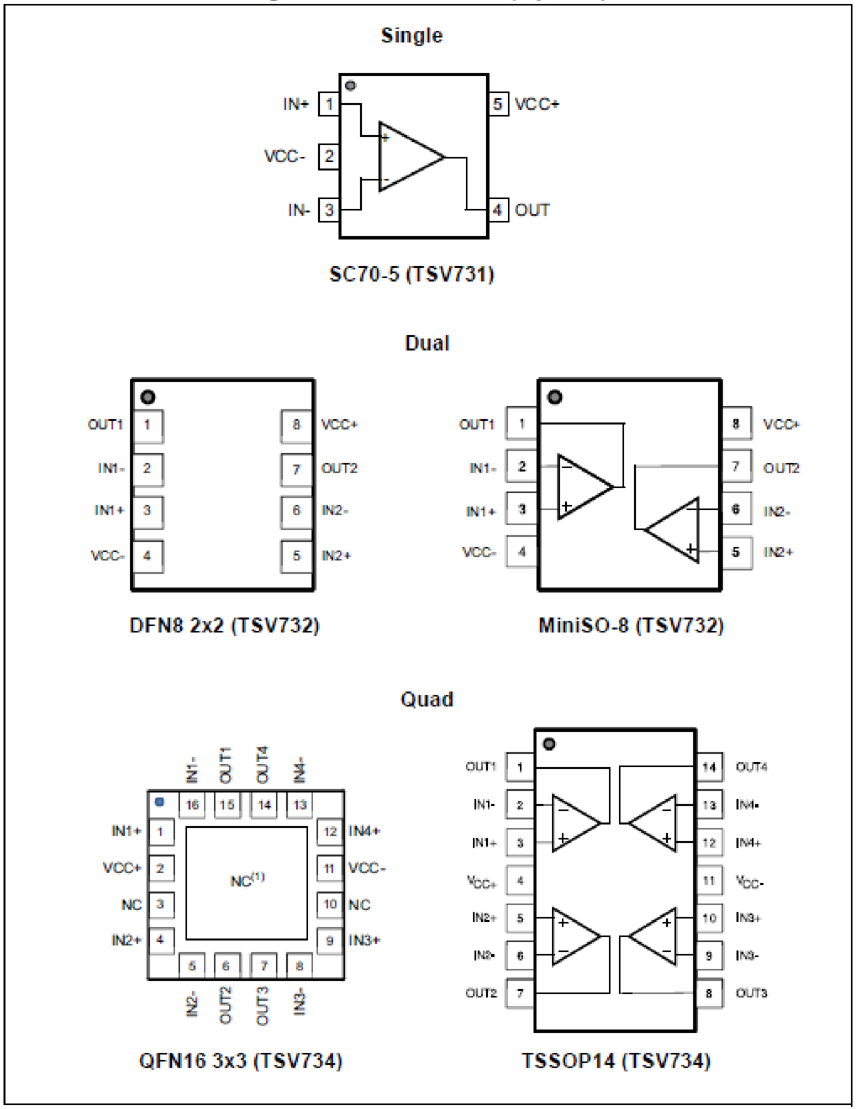
This is preliminary information on a new product now in development or undergoing evaluation. Details are subject to change without notice. www.st.com

Datasheet 9: Operational Amplifier Datasheet

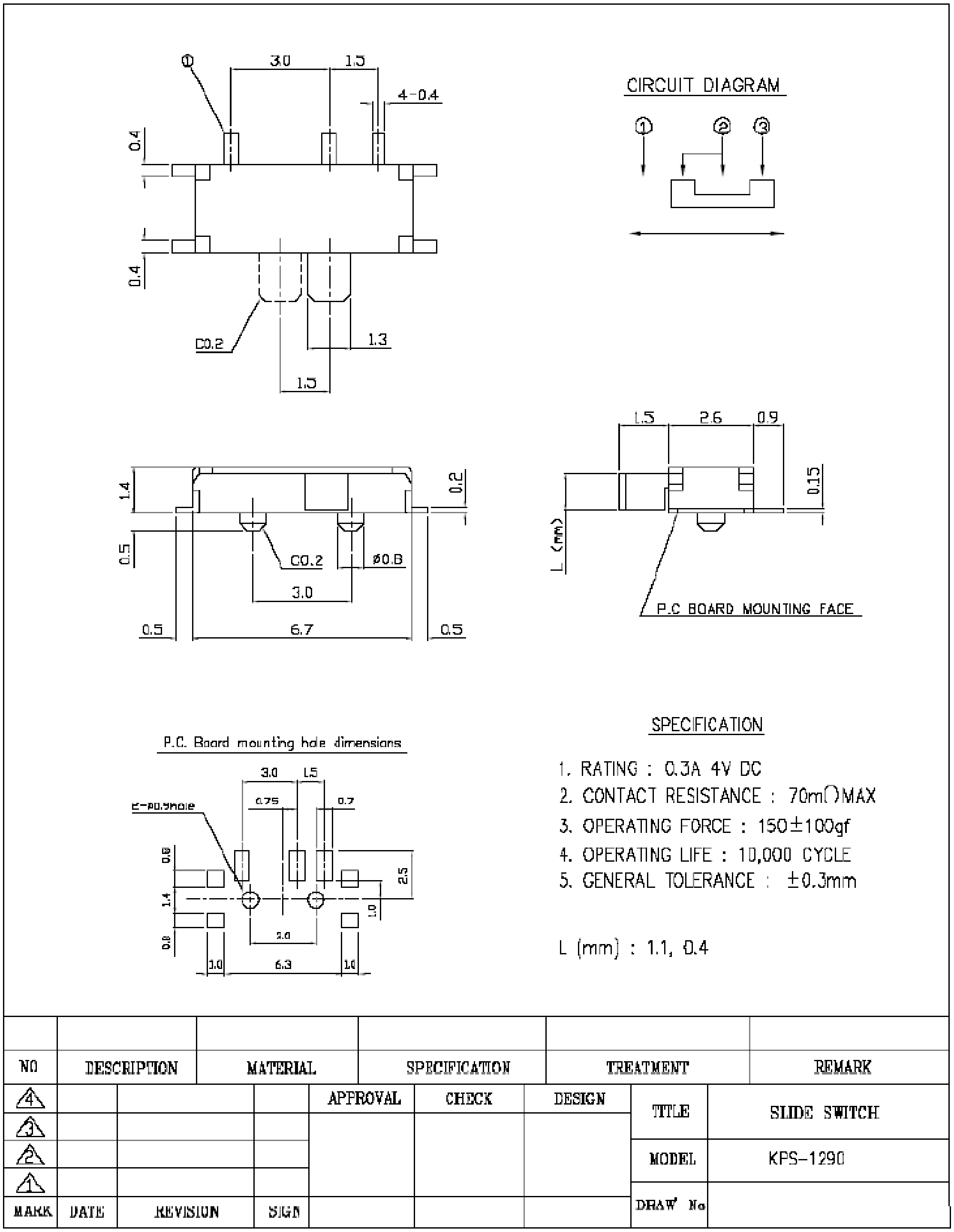
http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00065944.pdf

1 Pin connections

Figure 1. Pin connections (top view)



1. The exposed pads of the QFN16 3x3 can be connected to VCC- or left floating.



ON SHINE ENTERPRISE CO., LTD.

Datasheet 10: SMD Right Angle Switch Datasheet

http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Components/Switches/SLIDE.pdf

Shop | Learn | AVC

0 items | You are **not** logged in. log in / register

Tutorials | Videos | Classes | Wish Lists | Distributors | Support | Contact | search...

Home | *Tutorials* | PowerCell Quickstart Guide

PowerCell Quickstart Guide

by a1ronzo | September 13, 2012 | 11 comments

Skill Level: ★ Beginner

Overview

The Powercell board can serve many purposes. The board is a single cell LiPo battery charger, along with an efficient regulator, that can supply 3.3V or 5V power to your project. The Powercell can also be permanently connected to your system, so that you will never need to remove the battery from your project.

This guide will go over how to charge your battery and how to add power to your system, as well as cover some of the configuration options for the Powercell.

Requirements

Here are a list of supplies you will need if you want to use the full features of this board.

- Soldering Iron w/ Solder
- micro-B USB Cable
- Powercell
- single cell LiPo battery
- Arduino or any other system that needs regulated power
- Break Away Male Headers - Right Angle
- Jumper Wires Premium 6" M/F Pack of 10

What it Does

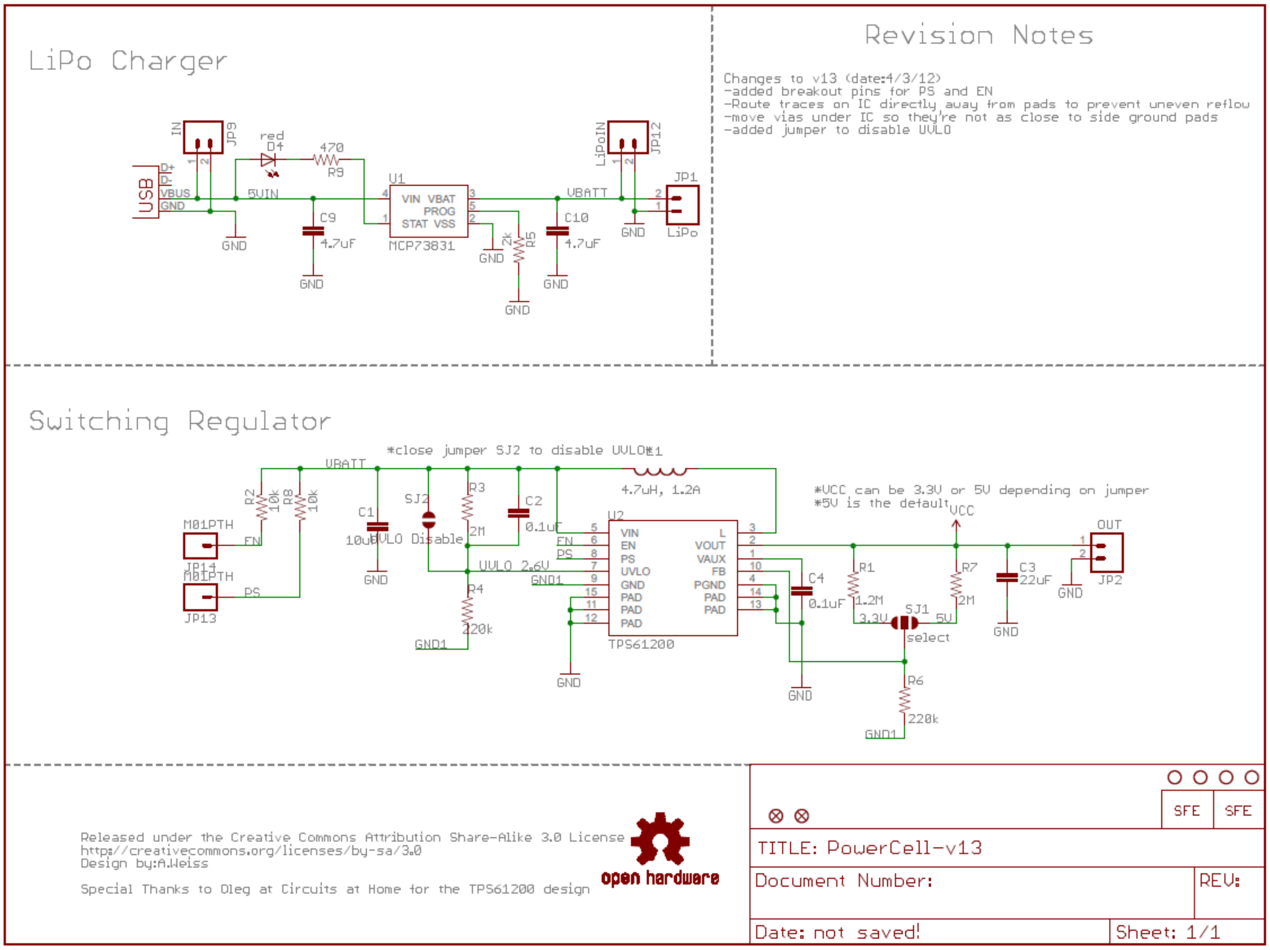
The Powecell can act alone as a single cell Lipo battery charger, it can act as a boost (5V) or buck (3.3V) regulator for your project, or it can perform both of these functions. *No soldering required, if you are only using the board to charge your battery over USB.*

Battery Charger:

The charger on the Powercell uses the MCP73831 from Microchip and can supply up to 500mA to a single cell LiPo battery. If you charge over the micro-B USB connector, the charge current will be 100mA. You can also externally charge your battery using the pins labeled 'charge' using your

Datasheet 11: Sparkfun PowerCell Quickstart Guide

https://www.sparkfun.com/tutorials/379



Datasheet 12: Sparkfun PowerCell Schematic

http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/PowerCell-v13.pdf muRata Capacitor data sheet

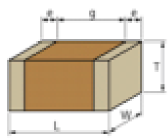
GRM155R71C104KA88#

" #" indicates a package specification code.

Available | Reflow OK | Complies with RoHS

< List of part numbers with package codes >
GRM155R71C104KA88D , GRM155R71C104KA88W , GRM155R71C104KA88J

Shape



L size	1.0 ±0.05mm
W size	0.5 ±0.05mm
T size	0.5 ±0.05mm
External terminal width e	0.15 to 0.35mm
Distance between external terminals g	0.3mm min.
Size code in inch(mm)	0402 (1005)

References

Packaging	Specifications	Minimum quantity
D	φ180mm Paper taping	10000
J	φ330mm Paper taping	50000
W	φ180mm Paper taping (W8P1*) *Width :8mm, Pocket pitch :1mm	20000

Mass (typ.)	
1 piece	1.6mg
φ180mm Reel	118g

Specifications

Capacitance	0.10μF ±10%
Rated voltage	16Vdc
Temperature characteristics (complied standard)	X7R(EIA)
Capacitance change rate	±15.0%
Temperature range of temperature characteristics	-55 to 125℃
Operating temperature range	-55 to 125℃

1 of 2

Attention

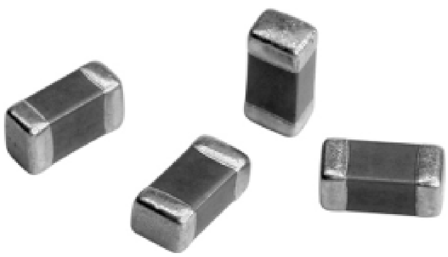
muRata Murata Manufacturing Co., Ltd. URL : http://www.murata.co.jp/

Last updated: 2014/03/09

Datasheet 13: 0.1µF Capacitor Datasheet

http://www.mouser.com/ds/2/281/product-235612.pdf

Surface Mount Multilayer Ceramic Chip Capacitors for Commodity Applications



FEATURES

- Available from 0402 to 1210 body sizes
- Ultra stable C0G (NP0) dielectric
- High capacitance in X5R, X7R, Y5V
- For high frequency applications
- Ni-barrier with 100 % tin terminations
- Dry sheet technology process
- Noble Metal Electrode system (NME): For certain C0G (NP0) values
- Base Metal Electrode system (BME): For X5R, X7R, Y5V and certain C0G (NP0) values
- Material categorization: For definitions of compliance please see www.vishay.com/doc?99912

RoHS COMPLIANT
HALOGEN FREE
GREEN (5-2008)

APPLICATIONS

- Consumer electronics
- Telecommunications
- Data processing
- Mobile applications

ELECTRICAL SPECIFICATIONS

Operating Temperature:
C0G (NP0): -55 °C to +125 °C
X5R: -55 °C to +85 °C
X7R: -55 °C to +125 °C
Y5V: -25 °C to +85 °C

Capacitance Range:
C0G (NP0): 0.5 pF to 39 nF
X5R: 47 nF to 100 µF
X7R: 100 pF to 47 µF
Y5V: 10 nF to 100 µF

Voltage Range:
C0G (NP0): 10 V_{DC} to 100 V_{DC}
X5R: 6.3 V_{DC} to 50 V_{DC}
X7R: 10 V_{DC} to 100 V_{DC}
Y5V: 6.3 V_{DC} to 100 V_{DC}

Temperature Coefficient of Capacitance (TCC):
C0G (NP0): 0 ppm/°C ± 30 ppm/°C from -55 °C to +125 °C
X5R: ± 15 % from -55 °C to +85 °C without voltage applied
X7R: ± 15 % from -55 °C to +125 °C without voltage applied
Y5V: +30 %/-80 % from -25 °C to +85 °C without voltage applied

Insulation Resistance (IR) at U_R:
≥ 10 GΩ or R x C ≥ 500 Ω x F whichever is less

Test Conditions for Capacitance Tolerance:
Preconditioning for X5R, X7R, Y5V MLCC: Perform a heat treatment at +150 °C ± 10 °C for 1 h, then leave in ambient condition for 24 h ± 2 h before measurement

Test Conditions for Capacitance and DF Measurement:
Measured at conditions of 30 % to 70 % related humidity.

C0G (NP0): Apply 1.0 V_{RMS} ± 0.2 V_{RMS}, 1.0 MHz ± 10 % for caps ≤ 1000 pF, at +25 °C ambient temperature
Apply 1.0 V_{RMS} ± 0.2 V_{RMS}, 1.0 kHz ± 10 % for caps > 1000 pF, at +25 °C ambient temperature

X5R/X7R: Caps ≤ 10 µF apply 1.0 V_{RMS} ± 0.2 V_{RMS}, 1.0 kHz ± 10 %, at +25 °C ambient temperature [(1)]
Caps > 10 µF apply 0.5 V_{RMS} ± 0.2 V_{RMS}, 120 Hz ± 20 %, at +25 °C ambient temperature

Y5V: Caps ≤ 10 µF apply 1.0 V_{RMS} ± 0.2 V_{RMS}, 1.0 kHz ± 10 %, at +20 °C ambient temperature
Caps > 10 µF apply 0.5 V_{RMS} ± 0.2 V_{RMS}, 120 Hz ± 20 %, at +20 °C ambient temperature

Note
(1) Test conditions: 0.5 V_{RMS} ± 0.2 V_{RMS}, 1 kHz ± 10 %
X7R: 0603: ≥ 2.2 µF/10 V
0805: 10 µF (6.3 V and 10 V)
X5R: 0402: ≥ 4.7 µF/6.3 V and ≥ 2.2 µF/10 V
0603: 10 µF/6.3 V

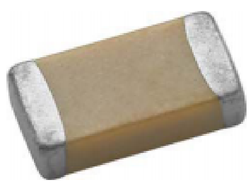
Aging Rate:
C0G (NP0): 0 % per decade
X5R: 6.3 V_{DC}/10 V_{DC}: 3 % maximum per decade
16 V_{DC}/25 V_{DC}: 2 % maximum per decade
X7R: ≤ 10 V_{DC}: 1.5 % maximum per decade
≥ 16 V_{DC}: 1 % maximum per decade
Y5V: 6.3 V_{DC}: 12.5 % maximum per decade
10 V_{DC}/16 V_{DC}: 9 % maximum per decade
≥ 25 V_{DC}: 7 % maximum per decade

Dielectric Strength Test:
This is the maximum voltage the capacitors are tested 1 s to 5 s period and the charge/discharge current does not exceed 50 mA.
≤ 100 V_{DC}: 250 % of rated voltage

Datasheet 14: 3.3nF Capacitor Datasheet

http://www.mouser.com/ds/2/427/vjw1bcbascomseries-223529.pdf

Surface Mount Multilayer Ceramic Chip Capacitors for High Frequency



FEATURES

- Case size 0402, 0603, 0805
- High frequency
- Ultra-stable dielectric material
- Lead (Pb)-free terminations code "X"
- Tin/lead termination code "L"
- Surface mount, wet build process
- Reliable Noble Metal Electrode (NME) system
- Made with a combination of design, materials and tight process control to achieve very high field reliability
- Material categorization: For definitions of compliance please see www.vishay.com/doc?99912

Available RoHS* Available HALOGEN FREE GREEN (5-2008) Available

Note

* This datasheet provides information about parts that are RoHS-compliant and/or parts that are non-RoHS-compliant. For example, parts with lead (Pb) terminations are not RoHS-compliant. Please see the information/tables in this datasheet for details.

APPLICATIONS

- Broadband wireless communication
- Satellite communication
- WiFi (802.11) and WiMax (802.16)
- VoIP networks and cellular base stations
- Subscriber based wireless devices

ELECTRICAL SPECIFICATIONS

Note

- Electrical characteristics at 25 °C unless otherwise specified

Operating Temperature: - 55 °C to + 125 °C

Capacitance Range:

0402: 1.0 pF to 82 pF

0603: 1.0 pF to 470 pF

0805: 1.0 pF to 1.5 nF

Voltage Rating: 25 V_{DC} to 250 V_{DC}

Temperature Coefficient of Capacitance (TCC):

C0G (D): 0 ppm/°C ± 30 ppm/°C from - 55 °C to + 125 °C with zero (0) V_{DC} applied

Dissipation Factor (DF):

C0G (D): 0.05 % max. at 1.0 V_{RMS} and 1 MHz for values ≤ 1000 pF

C0G (D): 0.05 % max. at 1.0 V_{RMS} and 1 kHz for values > 1000 pF

Aging Rate: 0 % maximum per decade

Insulation Resistance (IR):

At + 25 °C and rated voltage 100 000 MΩ minimum or 1000 ΩF, whichever is less

At + 125 °C and rated voltage 10 000 MΩ minimum or 100 ΩF, whichever is less

Dielectric Strength Test:

Performed per method 103 of EIA-198-2-E.

Applied test voltages:

≤ 200 V_{DC}-rated: Min. 250 % of rated voltage

> 200 V_{DC}-rated: Min. 200 % of rated voltage

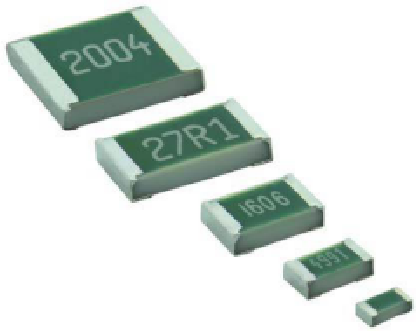
Datasheet 15: 20pF Capacitor Datasheet

http://www.mouser.com/ds/2/427/vjhifreq-109303.pdf

TNPW e3
Vishay

www.vishay.com

High Stability Thin Film Flat Chip Resistors



TNPW e3 precision thin film flat chip resistors are the perfect choice for most fields of modern electronics where highest reliability and stability is of major concern. Typical applications include test and measuring equipment, medical equipment, industrial, and automotive.

FEATURES

- Superior moisture resistivity (85 °C; 85 % RH)
- Excellent overall stability at different environmental conditions ≤ 0.05 % (1000 h rated power at 70 °C)
- AEC-Q200 qualified (sizes 0402 to 1206)
- Single lot date code (optional)
- Sulfur resistance verified according to ASTM B 809
- Material categorization: For definitions of compliance please see www.vishay.com/doc?99912

AUTOMOTIVE GRADE
RoHS COMPLIANT
HALOGEN FREE

APPLICATIONS

- Test and measuring equipment
- Medical equipment
- Industrial equipment
- Automotive

TECHNICAL SPECIFICATIONS					
DESCRIPTION	**TNPW0402 e3**	**TNPW0603 e3**	**TNPW0805 e3**	**TNPW1206 e3**	**TNPW1210 e3 (1)**
DIN size	0402	0603	0805	1206	1210
Metric size code	RR 1005M	RR 1608M	RR 2012M	RR 3216M	RR 3225M
Resistance range	10 Ω to 100 kΩ	10 Ω to 332 kΩ	10 Ω to 1 MΩ	10 Ω to 2 MΩ	10 Ω to 3.01 MΩ
Resistance tolerance	± 1 %; ± 0.5 %; ± 0.1 %				
Temperature coefficient	± 50 ppm/K; ± 25 ppm/K; ± 15 ppm/K; ± 10 ppm/K				
Rated dissipation, P_{70} (2)	0.063 W	0.1 W	0.125 W	0.25 W	0.33 W
Operating voltage, $U_{max.}$ AC_{RMS} or DC	50 V	75 V	150 V	200 V	200 V
Permissible film temperature, $\vartheta_{F\ max.}$ (2)	155 °C				
Operating temperature range	-55 °C to 125 °C (155 °C)				
Insulation voltage:					
1 min; U_{ins}	75 V	100 V	200 V	300 V	300 V
Continuous	75 V	75 V	75 V	75 V	75 V
Failure rate: $FIT_{observed}$	≤ 0.1 x 10^{-9}/h				

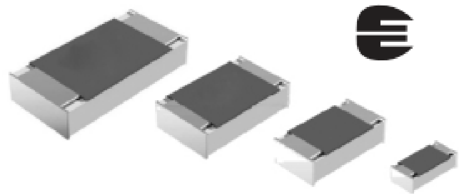
Notes
(1) Size not specified in EN 140401-801.
(2) Please refer to APPLICATION INFORMATION below.

Revision: 05-Feb-14 1 Document Number: 28758
For technical questions, contact: thinfilmchip@vishay.com

Datasheet 16: 10kΩ Datasheet

http://www.vishay.com/docs/28758/tnpw_e3.pdf

Professional Thin Film Chip Resistors



MCS 0402, MCT 0603, MCU 0805 and MCA 1206 professional thin film flat chip resistors are the perfect choice for most fields of modern professional electronics where reliability and stability is of major concern. Typical applications include telecommunication, medical equipment and high-end computer and audio/video electronics.

FEATURES

- Approved to EN 140401-801
- Excellent overall stability: Class 0.5
- Professional tolerance of value: ± 0.5 % and ± 1 %
- Lead (Pb)-free solder contacts
- Waste gas resistance verified by ASTM B 809
- Material categorization: For definitions of compliance please see www.vishay.com/doc?99912

RoHS COMPLIANT

APPLICATIONS

- Automotive
- Telecommunication
- Medical equipment
- Industrial equipment

TECHNICAL SPECIFICATIONS								
	MCS 0402		MCT 0603		MCU 0805		MCA 1206	
Imperial size	0402		0603		0805		1206	
Metric size code	RR1005M		RR1608M		RR2012M		RR3216M	
Resistance range	10 Ω to 4.99 MΩ; 0 Ω		1 Ω to 10 MΩ; 0 Ω		1 Ω to 10 MΩ; 0 Ω		1 Ω to 2 MΩ; 0 Ω	
Resistance tolerance	± 1 %; ± 0.5 %							
Temperature coefficient	± 50 ppm/K; ± 25 ppm/K							
Operation mode	Standard	Power	Standard	Power	Standard	Power	Standard	Power
Rated dissipation, P_{70} (1)	0.063 W	0.1 W	0.1 W	0.125 W	0.125 W	0.2 W	0.25 W	0.4 W
Operating voltage, $U_{max.}$ AC/DC	50 V		75 V		150 V		200 V	
Permissible film temperature, ϑ_F max.	125 °C	155 °C	125 °C	155 °C	125 °C	155 °C	125 °C	155 °C
Operating temperature range	- 55 °C to 125 °C	- 55 °C to 155 °C	- 55 °C to 125 °C	- 55 °C to 155 °C	- 55 °C to 125 °C	- 55 °C to 155 °C	- 55 °C to 125 °C	- 55 °C to 155 °C
Max. resistance change at P_{70} for resistance range, $\|\Delta R/R\|$ max., after:	10 Ω to 4.99 MΩ		1 Ω to 10 MΩ		1 Ω to 10 MΩ		1 Ω to 2 MΩ	
1000 h	≤ 0.25 %	≤ 0.5 %	≤ 0.25 %	≤ 0.5 %	≤ 0.25 %	≤ 0.5 %	≤ 0.25 %	≤ 0.5 %
8000 h	≤ 0.5 %	≤ 1.0 %	≤ 0.5 %	≤ 1.0 %	≤ 0.5 %	≤ 1.0 %	≤ 0.5 %	≤ 1.0 %
225 000 h	≤ 1.5 %	-	≤ 1.5 %	-	≤ 1.5 %	-	≤ 1.5 %	-
Insulation voltage:								
1 min; U_{ins}	75 V		100 V		200 V		300 V	
Continuous	75 V		75 V		75 V		75 V	
Failure rate: $FIT_{observed}$	≤ 0.1 x 10^{-9}/h		≤ 0.1 x 10^{-9}/h		≤ 0.1 x 10^{-9}/h		≤ 0.1 x 10^{-9}/h	

Notes

- These resistors do not feature a limited lifetime when operated within the permissible limits. However, resistance value drift increasing over operating time may result in exceeding a limit acceptable to the specific application, thereby establishing a functional lifetime.

(1) The power dissipation on the resistor generates a temperature rise against the local ambient, depending on the heat flow support of the printed-circuit board (thermal resistance). The rated dissipation applies only if the permitted film temperature is not exceeded.

Datasheet 17: 10MΩ Datasheet

http://www.vishay.com/docs/28705/mc_pro.pdf