Emory Periodic Limb Movement Monitoring System

ECE4012 Senior Design Project

Team Name: The Dream Team
Project Advisor: William Smith
Project Sponsor: Emory University Sleep Center

Mauricio Builes Zapata, mbuiles3@gatech.edu
Xavier Williams, xwilliams3@gatech.edu
James Grusy, jagrusy@gatech.edu
Isabel Anderson, ianderson7@gatech.edu

Submitted

12/14/2017

Table of Contents

Executive Summary

The Dream Team designed a Restless Leg Syndrome monitoring system for a more convenient and cost-effective way of detecting the disease in comparison with traditional methods. Restless Leg Syndrome is a sleep disorder characterized by periodic limb movements that incapacitate and deteriorate a person's sleep. In order to properly diagnose the disorder, a patient is sent to a sleep clinic where they are monitored by sleep clinicians while connected to different data collecting apparatuses to characterize their pathology. As expected, this creates discomfort in the patient which in itself affects their sleep and is costly and inefficient for both the doctors and the patient. With these constraints in mind and in conjunction with Emory's Sleep Center, The Dream Team designed and prototyped a lightweight, power efficient and wireless device to record leg movements for a week's time in the comfort of their own house.

The outcome of the team's work was an embedded solution that took advantage of wireless power transfer protocols, an accelerometer for motion recording, an antenna for Bluetooth data transfer in a 2.5 inch x 2 inch printed circuit board weighing less than 40 grams. The device is controlled by TI's CC2650 microcontroller unit, an ultra-low power consumption device with the added benefit of integrated Bluetooth functionality. Furthermore, the team designed and demonstrated a graphical user interface for the visualization of the collected data with the ability to annotate and classify time periods as per observations gathered from visits to the sleep clinic. In total, the cost per device totaled around USD $80, a small fraction of money in comparison to what going to a sleep center would be.

Emory Periodic Limb Movement Monitoring System

1. Introduction

The Dream Team set out to continue a multi semester project to build a sleep monitoring device, and was able to develop of an improved sleep monitoring device for use in Emory University's sleep research lab with the final cost analysis of $96.82. The device is to be worn on a patient's leg during sleep and is to collect Periodic Leg Movement (PLM) data. The data from the accelerometer is sent and stored to onboard storage (microSD card). Given more time for product development, the device would have the ability to wirelessly transmit the data collected to a nearby device (computer, smartphone, etc.) for processing and analysis.

1.1 Objective

The Dream Team decided to start a completely new device and GUI versus continuing with the progress made by the previous sleep monitoring team. The microcontroller software from the previous team was entirely redone to enhance the battery life, the Graphical User Interface (GUI) and data processing. In addition to these software enhancements, a PCB was designed for a more streamlined and lighter product. The addition of Qi wireless charging capabilities were added to the device which would effectively activate the Bluetooth module and automatically connect to a device to begin data transfer. The data is rendered in the designed GUI for the use and analysis by the physicians.

1.2 Motivation

This project is a joint effort between Emory University and Georgia Tech. The goal was to produce a sleep monitoring device that collects and later transmits data to be used in the diagnosis of sleep disorders related to PLMs (which are commonly linked to Restless Legs Syndrome) [1]. The device will be used by patients of Emory Sleep Clinic. Patients are to wear the device on their leg for up to five days for data collection. This project will improve upon the previous Georgia Tech senior design team's prototype. This will help overall to improve upon an obsolete device which the Emory Sleep Clinic currently uses: The PAM-RL. This device that Emory University currently uses is no longer on the market and costs approximately $800. The prototype designed by The Dream Team will cost a fraction of that: $379.00.

1.3 Background

The original sleep monitoring product that Emory used was the PAM-RL. This device, now obsolete, had cost and performance limitations that affected its usability. The performance issues related mainly to the data transfer and battery life. The previous team's battery only lasted for 72 hours and the data transfer took up to 37 minutes from device to computer.

To improve upon the previous prototype, the Dream Team aimed to add further improvements to the GUI and enhance the data rendering capabilities. The Dream Team used a new microcontroller called the CC2650 wireless MCU from Texas Instruments which decreased the size of the board as well as add functionality such as a built-in microSD card slot and built in accelerometer.

2. Project Description and Goals

The end goal of this project was to construct a functioning sleep monitor which is to be worn around a patient's leg. The entire monitoring system was configured into a compact, anklet format. Within the anklet's structure, there was and accelerometer amongst sensor which will detect motion of the limbs throughout the day and night. The limb acceleration data was supposed to be exported into a corresponding GUI with a user-friendly interface for the clinicians. The device is meant for use only by the clinicians at the Emory Sleep Clinic and will display the collected limb acceleration data in a graphical (time) format. A PCB was successfully designed and built. Testing could not be done on the PCB because the team was unable to learn how to program the PCB before the Expo occurred. Testing could not be done on wireless charging because the rechargeable batteries that were ordered never came in. The accelerometer was successfully interfaced to the microcontroller. A low-power/sleep mode was able to be made for the device. There was trouble interfacing the SD card to the CC2650. The team did not have enough time to finish the interfacing the SD card or getting the bluetooth to work to send data to a computer running the GUI. The GUI was able to render data via file upload, annotate rendered data and save/restore annotated data. As shown in Table 2, the estimated cost of the equipment needed to fabricate the monitor is $96.82. The main functional goals were the following:

- Data collection software of limb acceleration
- Collected data exportation via Bluetooth
- Wireless inductive charging
- Contained inside a discrete, sealed package
- Battery life of at least 5 days
- Contained on a PCB

3. Technical Specifications

TABLE 1

SPECIFICATIONS OF THE PLM MONITOR

Description	**Goal**	**Actual Value**
Weight	65 grams	40 grams
Dimensions	9 cm x 5 cm x 1.5cm	6.5 cm x 5 cm x 0.5 cm (without packaging)
Operating Time	>5 days	4 days, based on current consumption of .9 mA (untested with running firmware)
Sampling Rate	10 Hz	10 Hz
Download Time	<37 minutes	untested
Built on a PCB	Yes	Yes
GUI Display	Yes	Yes

Sealed and Waterproof	Yes	No (not developed)
Raw Data Access	Yes	Yes (CSV output)

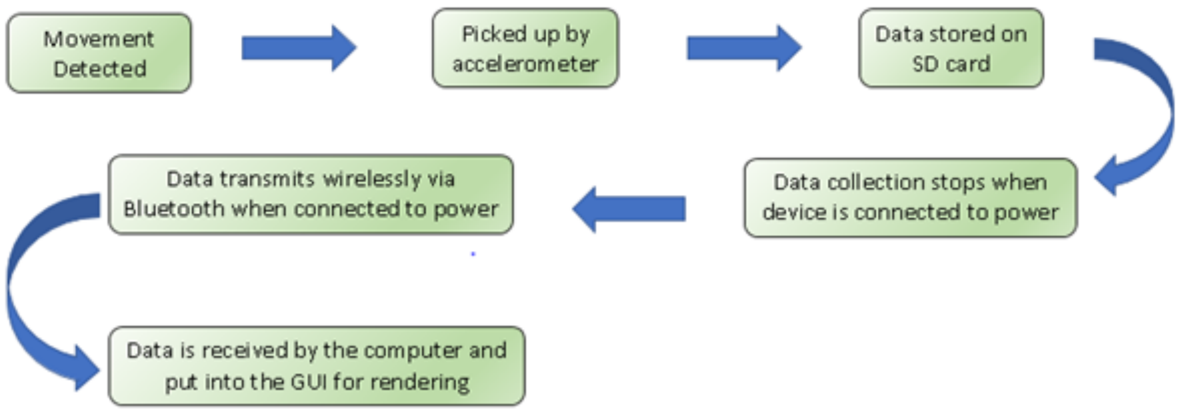
4. Design Approach and Details

4.1 <u>Design Approach</u>

The Dream Team worked on the 4th generation Restless Leg Syndrome monitoring device. Based on previous team's work, team members identified Bluetooth capability, wireless power transfer, accelerometer data collection and the use of a microcontroller as the necessary components for the desired functioning. The design approach for the team's prototype relied on three concurrent developments: firmware, hardware, and GUI. The work on these aspects of the design were interconnected as to develop a functional prototype, with the final system integration completing the design.

The hardware development consisted in the component selection and design of the printed circuit board. The cornerstone of the component selection and the posterior board design was the selection of the microcontroller unit. The team evaluated the microPython board for its ease of development and previous use in past teams. However, upon further examination it was discovered that the only available packaging for the microPython board was a development board, making its integration to a PCB largely incompatible. As such, the team moved on to TI's CC2650 32 bit microcontroller, due to its advertised low power consumption, small footprint, and integrated Bluetooth stack. A QFN packaging option with 48 pins and over 20 GPIOs, 2 SPI and I2C peripherals was selected for the design due to its added versatility. Based on this selection, the accelerometer (LIS3DH), SD card (required due to the low memory in the CC2650), power switch and battery were chosen. It was decided that due to the small footprint of the board components, it would be feasible to have a coin cell battery directly attached to the board, rather than externally as initially proposed. This, however, limited the charge capacity on the system, as rechargeable coin cell batteries are limited in this aspect, with the team's being rated at 85 mAh. Furthermore, it was the team's decision to add a copper trace antenna to the board in order to use Bluetooth's 2.4 GHz RF band. The team based the antenna design on a TI reference application note, picking an inverted F antenna and replicating in it on Eagle CAD software. With component selection finished, the printed circuit board was designed on an FR4 substrate with 2 layers of copper. Trace width for the RF connection was matched to the feed in the antenna, with the remaining traces being 6 mils and 15 mils for signals and power connections, respectively. Copper pours were used and tied to the ground signal to act as heat sinks, simplify routing, and minimize power supply noise. Placement of vias, decoupling capacitors, current regulating inductors, and oscillator crystals was determined based on reference designs and antenna design guides. In order to achieve wireless power transfer, as well as to regulate the charging current to the battery for safety reasons, the board contained a microUSB female adaptor and several input 'pins' (for both troubleshooting and current delivery). An external charging copper loop was store bought to facilitate design and reduce clutter on the PCB, but future developments could be made to integrate the inductive charging trace into the board. Charging was successful as current was observed into the board's power traces upon placement on top of a charging base. The system's schematic, board layout and bill of materials can be found in Appendix A for reference.

Firmware development happened simultaneously with the board design and was facilitated by using TI's Launchpad prototyping board. By adding an accelerometer and sd card holder to a breadboard, and powering the system through a USB connection, it was possible to simulate the foreseen final PCB in order to program the microcontroller unit. Development happened in C using TI's CCS IDE. The team took advantage of TI's Real Time Operating System to avoid developing the entire system from scratch and to take advantage of the provided interfaces with the peripherals. The data path structure can be seen in Figure 1.



Movement Detected
Picked up by accelerometer
Data stored on SD card
Data collection stops when device is connected to power
Data transmits wirelessly via Bluetooth when connected to power
Data is received by the computer and put into the GUI for rendering

Figure 1. Data path for movement data acquisition, storage, and delivery.

The LIS3DH accelerometer allowed for communication through both SPI and I2C. It was determined that using I2C was adequate due to its simpler implementation and the lack of necessity for the high speed of SPI given a 10 Hz sampling rate. This was completed successfully and properly demonstrated in the breadboard, with the accelerometer reporting 3 axis measurements up to 16g. The team ran into issues when communicating with the SD card. Even though the SPI interface worked on the microcontroller, the team could not get the file system properly configured in the SD card through SPI communications. TI did not have a driver for interfacing with SD cards for this particular microcontroller. There were various post about interfacing SD cards to the CC2650 on TI's help forum for the device, however there had not been any successful attempts from any of the post. Various examples were looked through to learn how a SD driver could be written. Out of the examples, TI’s MSP had the best example of how to write an SD driver. Toward the beginning of the semester the team tried to port the driver from the MSP to the CC2650. However, there were issue in porting the driver due to the fact that the power and clock protocols differed vastly between each device. In order to accurately port the SD driver to the CC2650, some of the files in th RTOS would have to be changed to allow for the inclusion of the hardware attributes in the code. The team attempted this method, but was unsuccessful. Another attempt that the team tried was making a bit-banged SD driver, using the SPI driver that was provided by TI. In order to make a bit-banged version of the driver commands would need to be sent to the SD card using SPI. The difficulty was correctly implementing the commands and sending them to get a correct response back from the SD card. The team ran out of time with this roadblock, determining that future work must be implemented to resolve this issue if the device is to keep being worked on. Upon closer examination, it was found that an SPI port was improperly mapped, hence creating the issue. Further, working on the Bluetooth communications is still to be implemented in the future.

The graphical user interface was based on what was observed on a visit to the Emory Sleep Center and from feedback from doctors and sleep technicians. The goal of the GUI was to portray an entire night’s of data with the ability to zoom, pan, annotate the data, as well as to export and manipulate the data with external filters. As explained earlier, this development was done in parallel with the firmware and hardware development and was completed using the Node.js environment. Using sample data it was possible to develop a graph GUI that displayed the entire length of the data with zooming capabilities, the three axis of movement and the possibility to label periods of time with text and comments, as per requested by sleep doctors. A demonstration to Emory revealed feedback for future improvements. Further development will should include a way to show multiple nights on a single page, more customizations for window size, and possibly connection to a database for historical data trends. An image of the current graphical user interface can be seen in Figure 2.

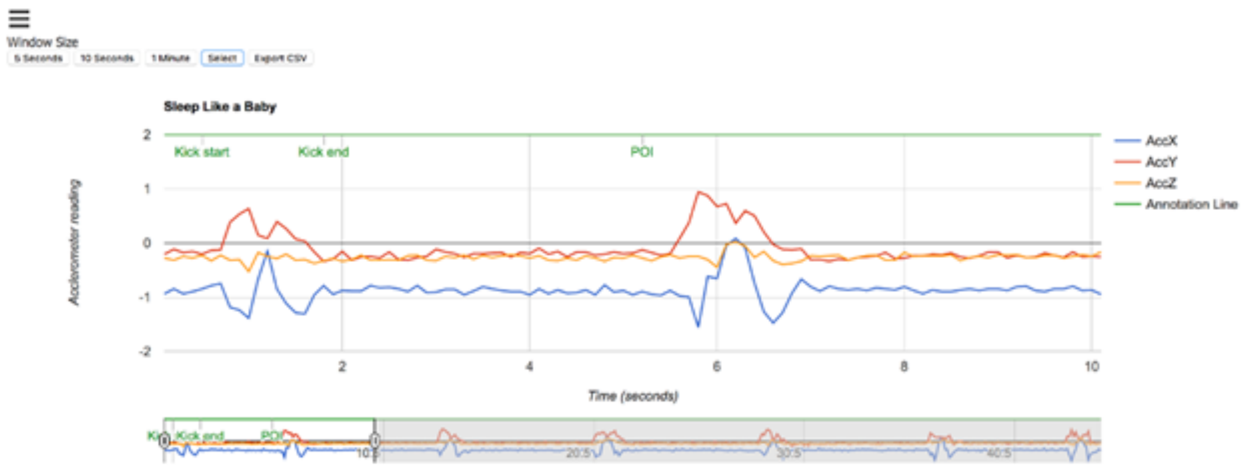


Figure 2. Screenshot of accelerometer data showcasing pan, zoom, and annotation capabilities.

Finally, the team lacked the time to work on an enclosure, recognizing that in the future a team could develop and design a sealed and waterproof enclosure for the device.

4.2 <u>Codes and Standards</u>

- · Bluetooth Wireless Technology Standard: Communication standard that operates at 2.4 GHz bandwidth. Necessary for obtaining the data from the monitoring device into a computer for interfacing and data processing.
    - o Affected our device design by requiring a 2.4 GHz antenna. This resulted in an copper trace inverted F antenna being included to the PCB design.

- · Qi Wireless standard: Inductive charging standard being used in the smartphone industry. It is a powerful standard to have due to extensive documentation and proven viability. Will be used due to the monitoring device's necessity to be a fully enclosed product [4].
  - Was visible in the team's design by the inclusion of a female microUSB port as well as the external inductor coil acting as a power receiver, soldered to the device for demonstration. A Samsung charging base was used to prove charging.
- · The IEEE P360 Standard on wearable technology is a current working project by the Wearable Working Group that outlines basic safety and suitableness to wear wearable devices. It provides with technical requirements such as battery duration and optimization that help provide for guidelines in the project's technical considerations [5].
- · FCC regulations on class B devices: No radio interference is allowed to happen within a 10 meter radius. Given that the microcontroller and the Bluetooth module are commercially bought, they are already compliant with all necessary regulations [6].
- · HIPAA Regulations: Require all information to be encrypted when concerning patient health records [7].
  - Encryption of the team's data was omitted for prototyping purposes.

4.3 <u>Constraints, Alternatives, and Tradeoffs</u>

The design’s main constraint with respect to hardware revolved around the size of the printed circuit board. A pre-established size of 2.5 inches x 2 inches was selected, and the component selection and further board routing was limited to this size. Given the small footprint of all the integrated circuits and i/o components, this proved to be an acceptable and successful size.

The team had to make the decision between a larger capacity battery with a larger footprint or a smaller coin cell battery that could have been attached directly to the printed circuit board. The decision to use the smaller RJD2032 rechargeable battery stemmed from the fact that the consumption in the CC2650 was small enough to allow for the 85mAh to be enough.

As previously stated, the team opted for the CC2650 microcontroller unit made by Texas Instruments instead of the MicroPython pyBoard. By doing this, the team sacrificed ease of use and rapid implementation for a more robust, powerful, and more energy efficient device with a smaller footprint. At the time of the decision, this was deemed the correct decision. However, given that TI’s microcontroller was brand new, it lacked the proper documentation to allow the team to rapidly prototype and test it’s firmware, spending most of the time in debugging and configuring the system rather than implementing the features that it needed. A better alternative, from the team’s perspective, would have been to pick the CC2640, a similar device to the CC2650 but with vast documentation, reference designs and code examples. This device also contains an in chip Bluetooth stack, providing the same versatility that drew the team to pick the CC2650 in the first place.

5. Schedule, Tasks, and Milestones

The main tasks in this project were divided into three concurrent streams focusing on the PCB, microcontroller, and GUI development. There was overlap between the three teams as the designs for each stream closely interacted with the other streams. It's important that everyone understood the project from a hardware and software standpoint before the system miniaturization PCB design phase begins. The overall project timeline with the specific tasks, milestones, and critical paths is shown as both a GANTT chart and PERT chart in Appendix B. For the most part our team was able to follow the schedule except there was no time spent on pattern recognition and the testing/debugging stage was shorter than predicted due to time constraints. There was no opportunity to fully test the device in a clinical environment. The PERT displays the critical path with optimistic and pessimistic estimates for the time required to complete each task. Similar to the GANTT chart, the PERT chart accurately shows the three concurrent work streams, minus the pattern recognition.

6. Project Demonstration

The Sleep Monitor prototype progress was displayed at the Capstone Exposition. With the device not completed in its entirety, a demo (MCU and accelerometer) was displayed to demonstrate and mimic data collection which may be typical for a subject with sporadic movements. The accelerometer data in the x, y, and z -planes were displayed in live time to demonstrate the communications with the accelerometer, the data collection from any movements of the accelerometer, as well as the rendering software associated with this data collection. The PCB was displayed with all components soldered onto it. An extra PCB was displayed to show the size, weight and routing of the device. The GUI was on display to demonstrate the user interface, data collection and rendering software. A set of PLM data points was uploaded to the GUI in the form of a .csv file to allow the demonstration and manipulation of the data logging software. Within the GUI, the physician has the capability to select snippets of the graph anywhere throughout the data, whether it be a single point of interest or a span of time, label said data for later reference, and change viewing time scopes and scales. The GUI is a web based entity coded in js.node making it accessible remotely through any browser. With our rechargeable battery having not arrived in time for the Capstone Exposition, the battery life could not be tested. However, during the testing of the wireless charging, a current of 0.01 mA was measured across the microcontroller, thus confirming that the Qi charging was successful and the board could be powered on. Talking to the Bluetooth transmitter and microSD card proved to be difficult and was not accomplished in the span of time allotted for the project.

7. Marketing and Cost Analysis

7.1 Marketing Analysis

The target market will be patients of the Emory sleep clinic and any other sleep clinics who currently use the PAM-RL monitoring device. The PAM-RL monitoring device is currently the only comparable device on the market and it is no longer in production. The device proposed will be sold for $379 which is a fraction of the cost of $800 cost of the PAM-RL. In addition, the proposed device will have higher performance specifications with faster data transfer and better battery life.

7.2 Cost Analysis

The total cost for the parts will be approximately $96.82. Table 2 shows a breakdown of the components that will be used in the prototype.

TABLE 2

EQUIPMENT COST.

Product Description	Quantity	Unit Price ($)	Total Price ($)
MicroSD card 32GB [9]	1	$15.99	$15.99
LIS3DH Accelerometer	1	$1.53	$1.53
Custom Blank Printed Circuit Board	1	$33.00	$33.00
24 Mhz Oscillator	1	$0.53	$0.53
32.78 kHz Oscillator	1	$0.77	$0.77
Rechargeable Battery	1	$10.13	$10.13
.1 uF Capacitor	5	$0.10	$0.50
10 uF Capacitor	3	$0.02	$0.06
12 pF Capacitor	3	$0.12	$0.36
6.8 pF Capacitor	2	$0.01	$0.02
1 pF Capacitor	2	$0.05	$0.10
Power Switch	1	$0.62	$0.62
MicroUSB F	1	$1.00	$1.00
uSD Card Reader	1	$1.92	$1.92
10 uH Inductor	1	$0.03	$0.03
2.4 nH Inductor	2	$0.10	$0.20
2 nH Inductor	2	$0.09	$0.18
10 kOhms Resistor	3	$0.01	$0.03
33 Ohm Resistor	1	$0.01	$0.01
CC2650 - RGZ MCU	1	$4.90	$4.90
Battery Holder	1	$0.95	$0.95
Qi Wireless Charging Pad	1	$13.99	$13.99
Qi Wireless Receiver	1	$10.00	$10.00

Total Cost ($)			**$96.82**

The development costs will be approximately $9,896.82. The number of labor hours reflect the work that was done throughout the semester. The labor cost was calculated assuming an entry level engineer salary of $40 per hour. There were various software tasks that consumed 120 hours in total. PCB design took a total of 60 hours. Some of the biggest challenges were designing the PCB and trying to interface the uSD card reader to the CC2650.

TABLE 3

DEVELOPMENT COST.

Project Component	**Labor Hours**	**Labor Cost**	**Part Cost**	**Total Component Costs**
Software Development				
Embedded Development (Xavier)	45	$1,800.00		$1,800.00
GUI Development (James)	50	$2,000.00		$2,000.00
Sleep Mode (Xavier)	5	$200.00		$200.00
Testing and Debugging (James & Xavier)	20	$1,200.00		$1,200.00
Hardware Development			$96.82	$98.40
PCB Design (Mauricio & Isabel)	60	$2,400.00		$2,400.00
Testing and Debugging (Mauricio & Isabel)	10	$400.00		$400.00
Soldering (Mauricio & Isabel)	15	$600.00		$600.00
Team Meetings	20	$800.00		$800.00
Demo Preparation	20	$800.00		$800.00
Total Labor Costs	245	$9,800.00		
Total Part Costs			$96.82	
Total Cost ($)				**$9,896.82**

The total development costs were calculated using a value of 30% for fringe benefits and using 120% for overhead of material, labor and fringe. Table 4 shows the total development costs to be $41,170.78.

TABLE 4

TOTAL DEVELOPMENT COSTS.

Parts	**$96.82**
Labor	**$9,800.00**
Fringe Benefits, % of Labor	$2969.05
Subtotal	$12,865.87
Overhead, % of Material, Labor, and Fringe	$15,439.04
Total	**$41,170.78**

Assuming a production of 5,000 units, Table 5 shows a breakdown of what the selling price and profit per unit of the device would be. The assembly and testing labor assumes a technician works for 30 minutes assembling the device and 30 minutes testing the device at a pay rate of $20 per hour. The fringe benefits of labor assume the 30% figure used in Table 4. Likewise, the overhead is calculated with the same 120% figure used in Table 4. The sales expense will cover the cost to advertise the product and will represent 4% of the selling price. At a price of $379 per unit, 5,000 units of sale will generate $1,895,000 of revenue. This translates to a profit of $83.64 per unit and $418,200 overall profit.

TABLE 5

SELLING PRICE AND PROFIT PER UNIT
(BASED ON 5,000 UNIT PRODUCTION).

Parts Cost	**$96.82**
Assembly Labor	**$10.00**
Testing Labor	**$10.00**
Total Labor	$20.00
Fringe Benefits, % of Labor	$6.00

Subtotal	$122.82
Overhead, 120% of Material, Labor, and Fringe	$147.38
Subtotal, Input Costs	$270.20
Sales Expense	$15.16
Amortized Development Costs	$10.00
Subtotal, All Costs	$295.36
Profit	$83.64
Selling Price	**$379.00**

8. Conclusion

Our team set out to create a sleep monitoring device that could be worn for about a week, be small enough to be worn comfortably during sleep, and provide useful data for Emory's sleep clinicians to analyze. To attain these goals, our team decided to use the TI CC2650 to enable the power efficiency as well as build a PCB to enable a small form factor. In addition, our team designed a GUI to render this data for clinicians to analyze and annotate.

Currently the PCB is designed and printed, however it has not been tested due to the difficulty in programming the device as well as the battery not arriving in time. The microcontroller is able to interface with the accelerometer, however, it is not able to store the data collected due to numerous issues in communicating with the SD card. The GUI is able to render data via file upload and can zoom, pan, scroll, and annotate the data. There are further improvements that should be made based off of feedback from Emory. These improvements include a way to show multiple nights on a single page, more customizations for window size, and possibly connection to a database for historical data trends.

If we were to start this project again there are a few things we would change. The biggest change would be to use the TI CC2640 microcontroller. This microcontroller is slightly older the the CC2650, but offers similar specifications as well as a more mature development environment and library selection. The microcontroller development was by far the team's biggest hurdle. This is a great example of the tradeoff between working with the latest and greatest technology and working with something that's been widely used before. It is important to carefully consider these design constraints at the beginning of a project. While the superior finished product should utilize the latest technology available, it may be more feasible to utilize something that will allow for faster development based on your timeline and project scope.

9. Leadership Roles

Team Leader: Mauricio Builes Zapata

Lead Backend Developer: Alan Grusy

Lead Frontend Developer: Xavier Williams

Lead Hardware Integrator: Isabel Anderson

Webmaster: Alan Grusy

Expo Coordinator: Mauricio Builes Zapata

Communications Chair: Xavier Williams

Documentation: Isabel Anderson

10. References

[1] W. Blahd, "Periodic Limb Movement Disorder," WebMD, May 2016. [Online]. Available: http://www.webmd.com/sleep-disorders/periodic-limb-movement-disorder#1

[2] H. Hong, I. Lee, K. Lee, S. Meah, N. Ojong, and M. Son, "Emory Periodic Limb Movement Monitoring System," Five Year Journey, 21 November 2016.

[3] "Bluefruit LE - Bluetooth Low Energy," Adafruit, [Online]. Available: https://www.adafruit.com/product/1697. [Accessed 13 April 2017].

[4] "Universal Qi Wireless Charging Transmitter," Adafruit, [Online]. Available: https://www.adafruit.com/product/2162. [Accessed 13 April 2017].

[5] Consumer Electronics Society, "Standard For Wearable Consumer Devices," IEEE SA, [Online]. Avaliable: https://standards.ieee.org/develop/project/360.html

[6] Office of Engineering and Technology Federal Communications Commission, "Understanding The FCC Regulations," February 1996. [Online]. Available: https://transition.fcc.gov/bureaus/oet/info/documents/bulletins/oet62/oet62rev.pdf. [Accessed 15 November 2016].

[7] Office for Civil Rights, "Guidance to Render Unsecured Protected Health Information Unusable, Unreadable, or Indecipherable to Unauthorized Individuals," U.S Department of Health and Human Services, [Online]. Available: http://www.hhs.gov/hipaa/for-professionals/breach-notification/guidance/index.html. [Access 13 April 2017].

[8] "MicroPython pyboard lite v1.0 with accelerometer and headers," MicroPython, [Online]. Available: https://store.micropython.org/#/products/PYBLITEv1_0-ACH. [Accessed 13 April 2017].

[9] "SanDisk Memory Card," Best Buy, [Online]. Available: http://www.bestbuy.com/site/sandisk-ultra-plus-32gb-microsdhc-class-10-uhs-1-memory-card-gray-red/3142635.p?skuId=3142635. [Accessed 13 April 2017].

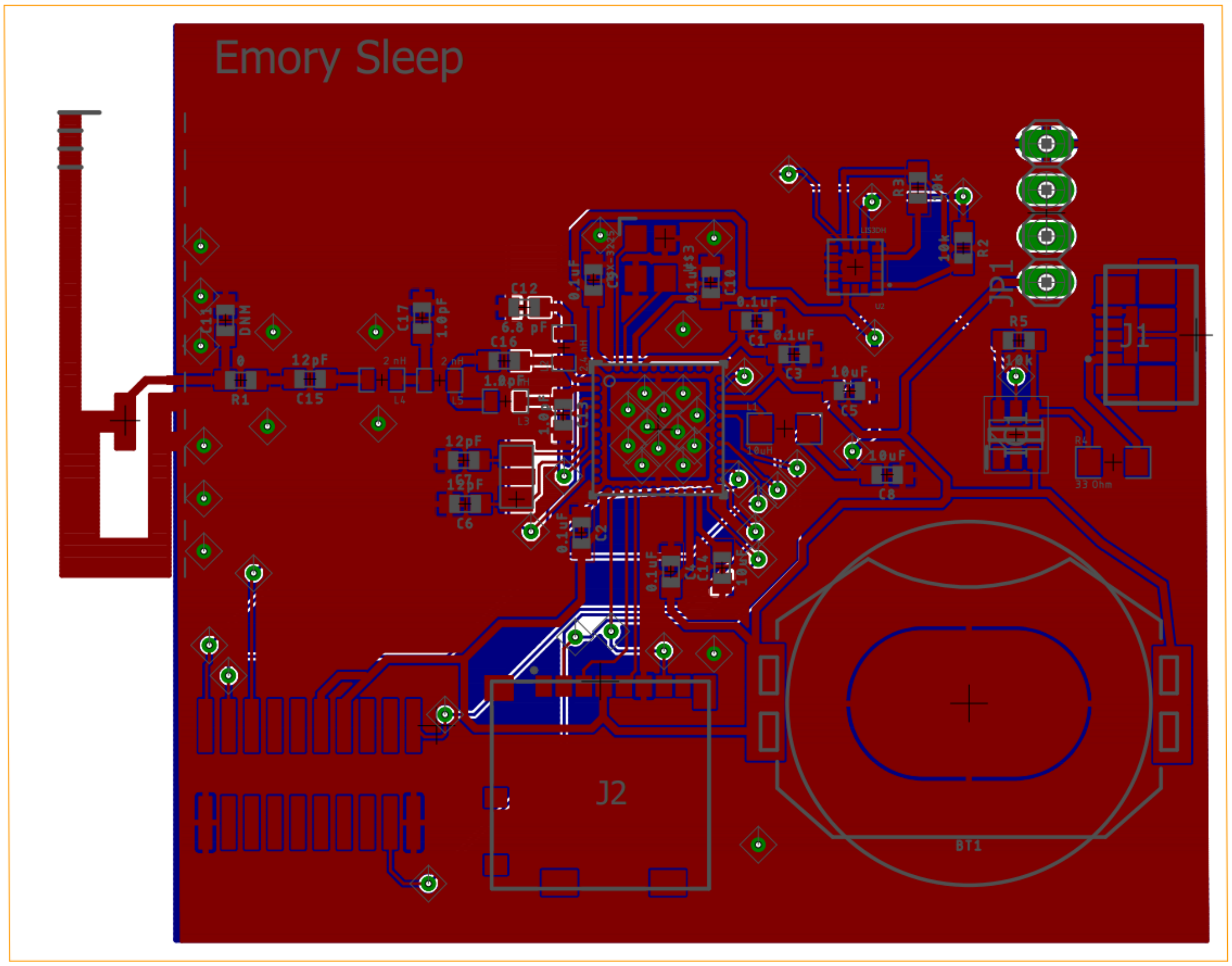
https://www.digikey.com/product-detail/en/stmicroelectronics/LIS3DHTR/497-10613-1-ND/2334355 - LIS3DH Accelerometer

https://www.mouser.com/ProductDetail/Epson-Timing/TSX-3225-240000MF15X-AC3/?qs=tC2qD9afNgpYJ9VgPl1%2FUg%3D%3D - Bluetooth crystal https://www.digikey.com/product-detail/en/epson/FC-135-32.7680KA-AC3/SER4085CT-ND/6132705?WT.srch=1&gclid=Cj0KCQjwsNfOBRCWARIsAGITapbuZ6ffIlyx_S7jc35-nIb5heAvdh3VbA8JqRWKFZs0Jv9_zLLlRi0aAr2YEALw_wcB - Clock Crystal

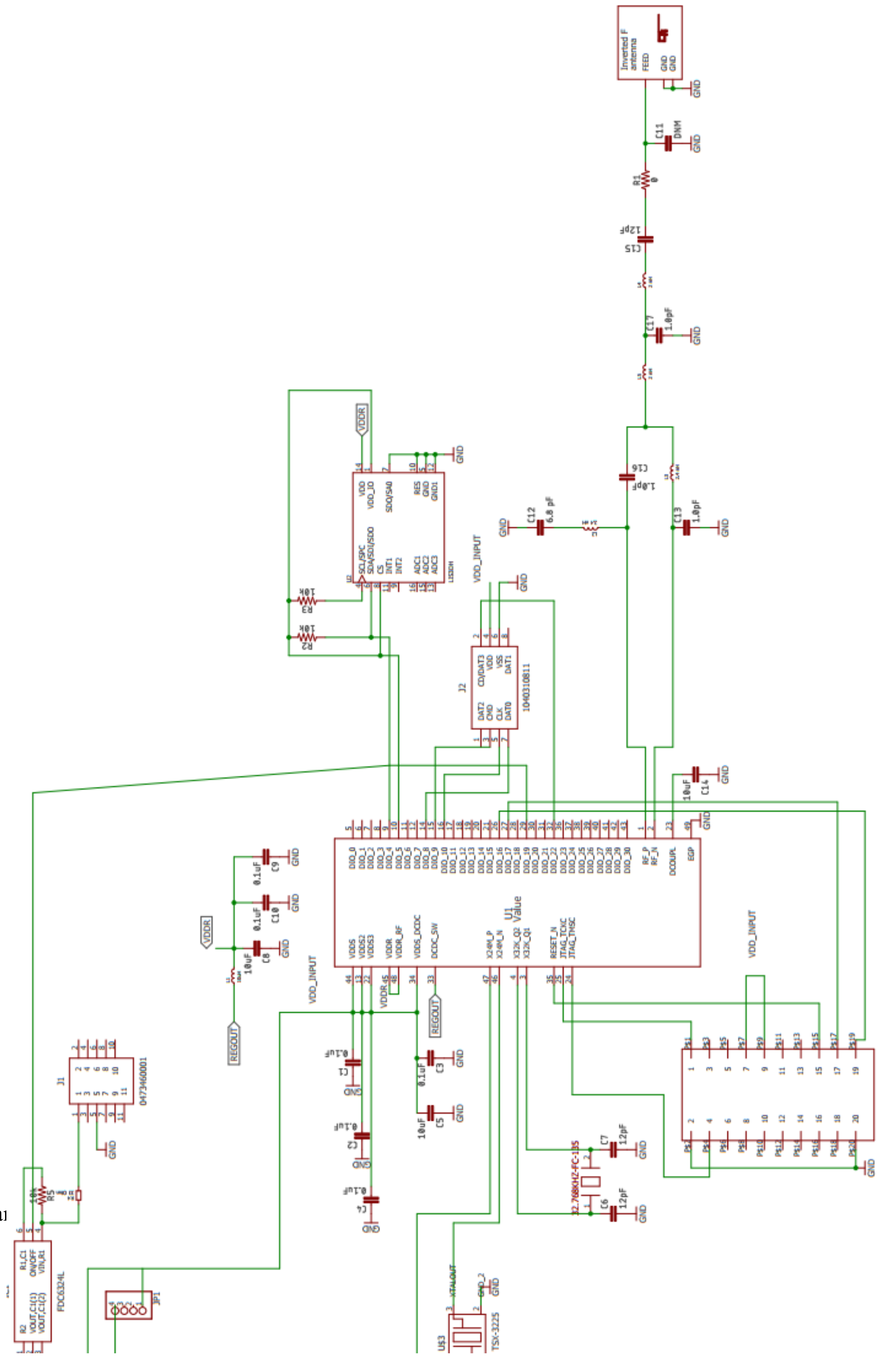
https://www.digikey.com/products/en?mpart=RJD2032C1&v=1572 - Rechargeable Battery

https://www.walmart.com/ip/Fast-Qi-Wireless-Charger-Charging-Pad-1000Mah-For-Samsung-Galaxy-White/144171775#read-more - wireless charging pad

Appendix A



Emory Sleep
R1
C15
C11
C17
C12
C16
C13
C6
C7
C2
C4
C14
C10
C1
C3
C5
C8
R2
R3
R4
R5
JP1
J1
J2
BT1



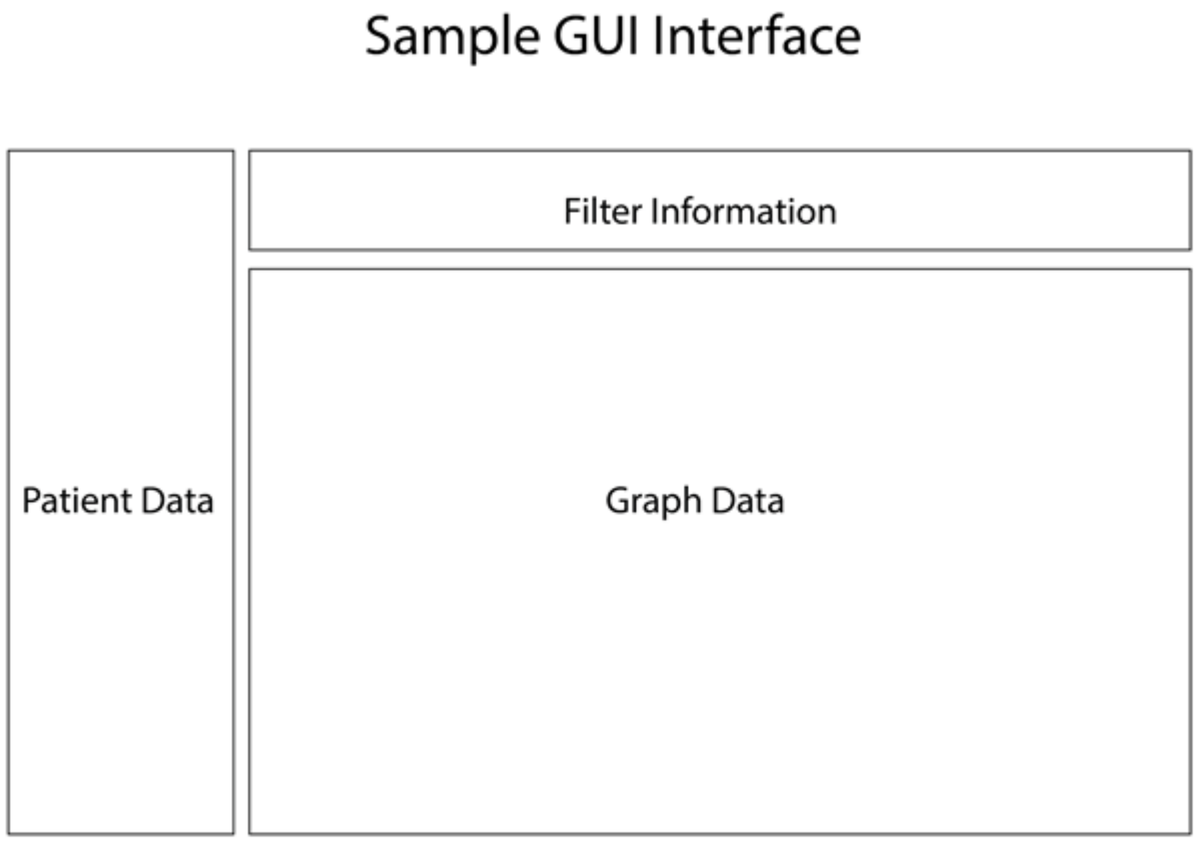
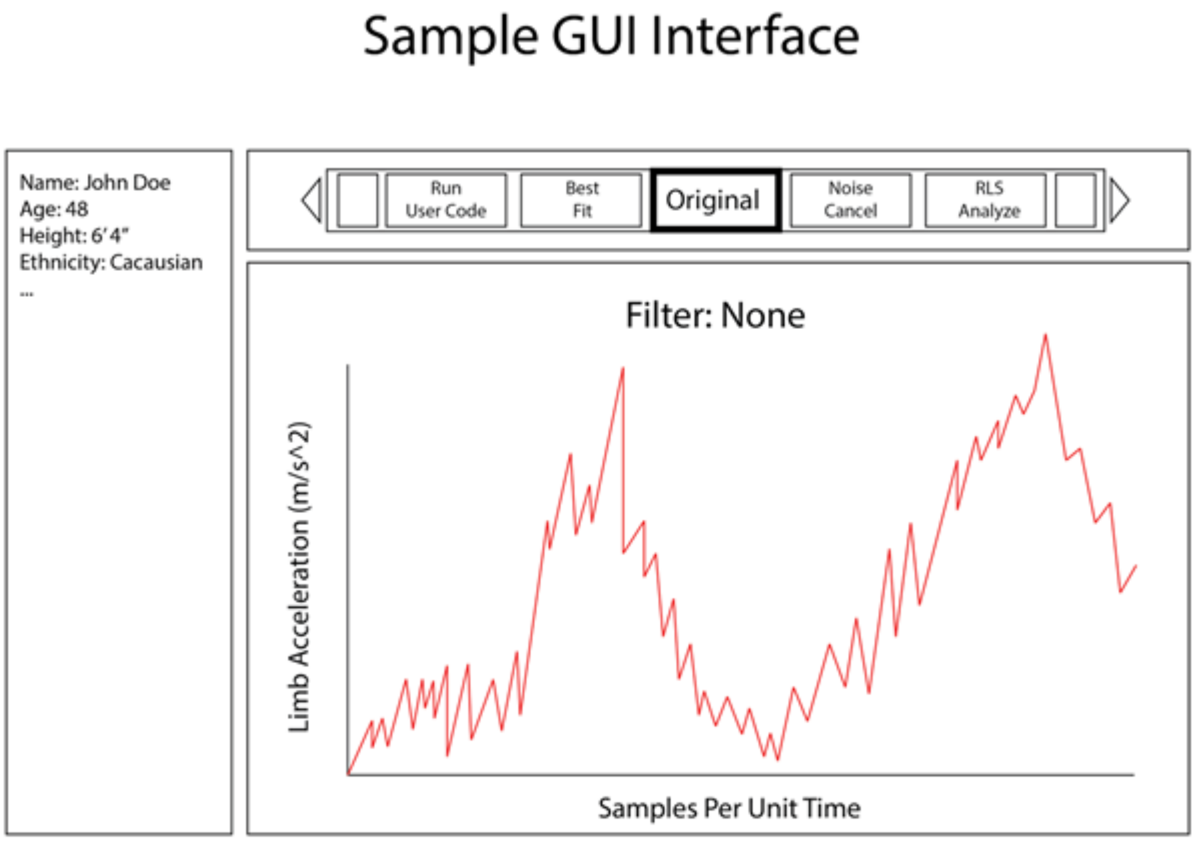


Figure 2. Sample GUI layout.[2]



Name: John Doe
Age: 48
Height: 6'4"
Ethnicity: Cacausian
...

Figure 3. Sample GUI interface with sample data.[2]

Appendix B

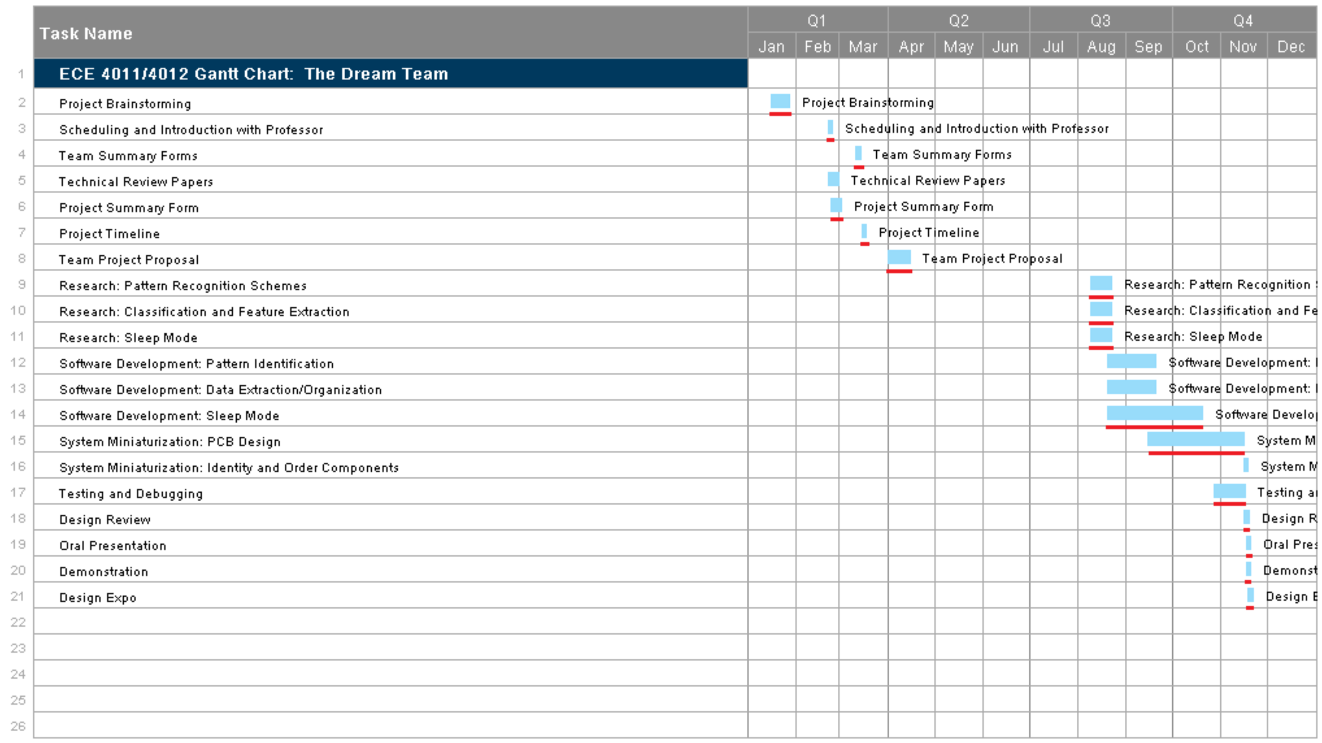


Figure 4. The project schedule in the form of a GANTT chart, the critical path is underlined in red.

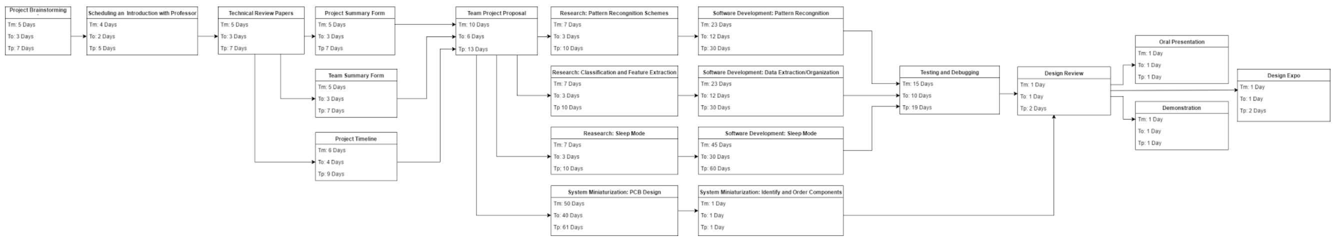
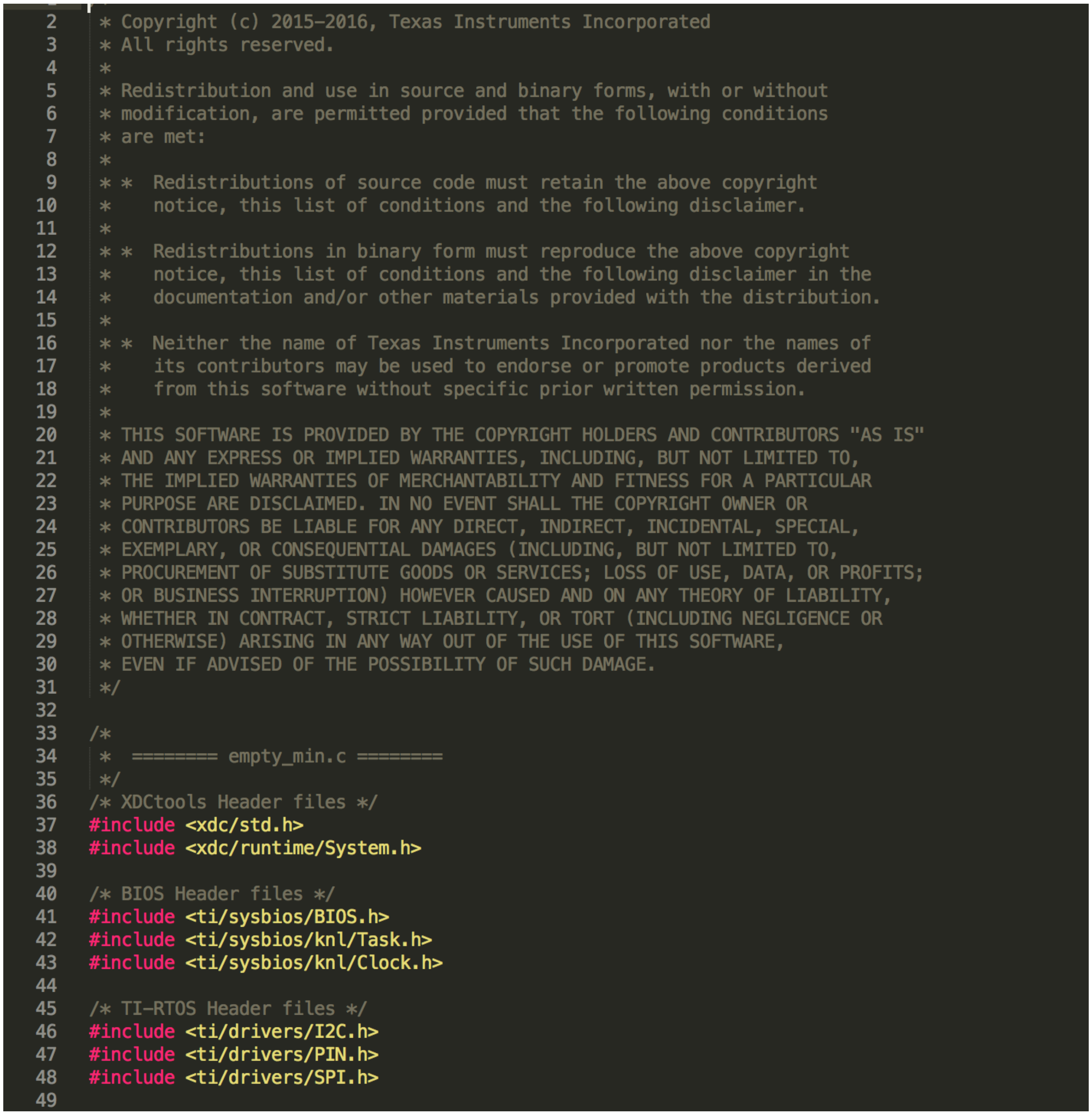


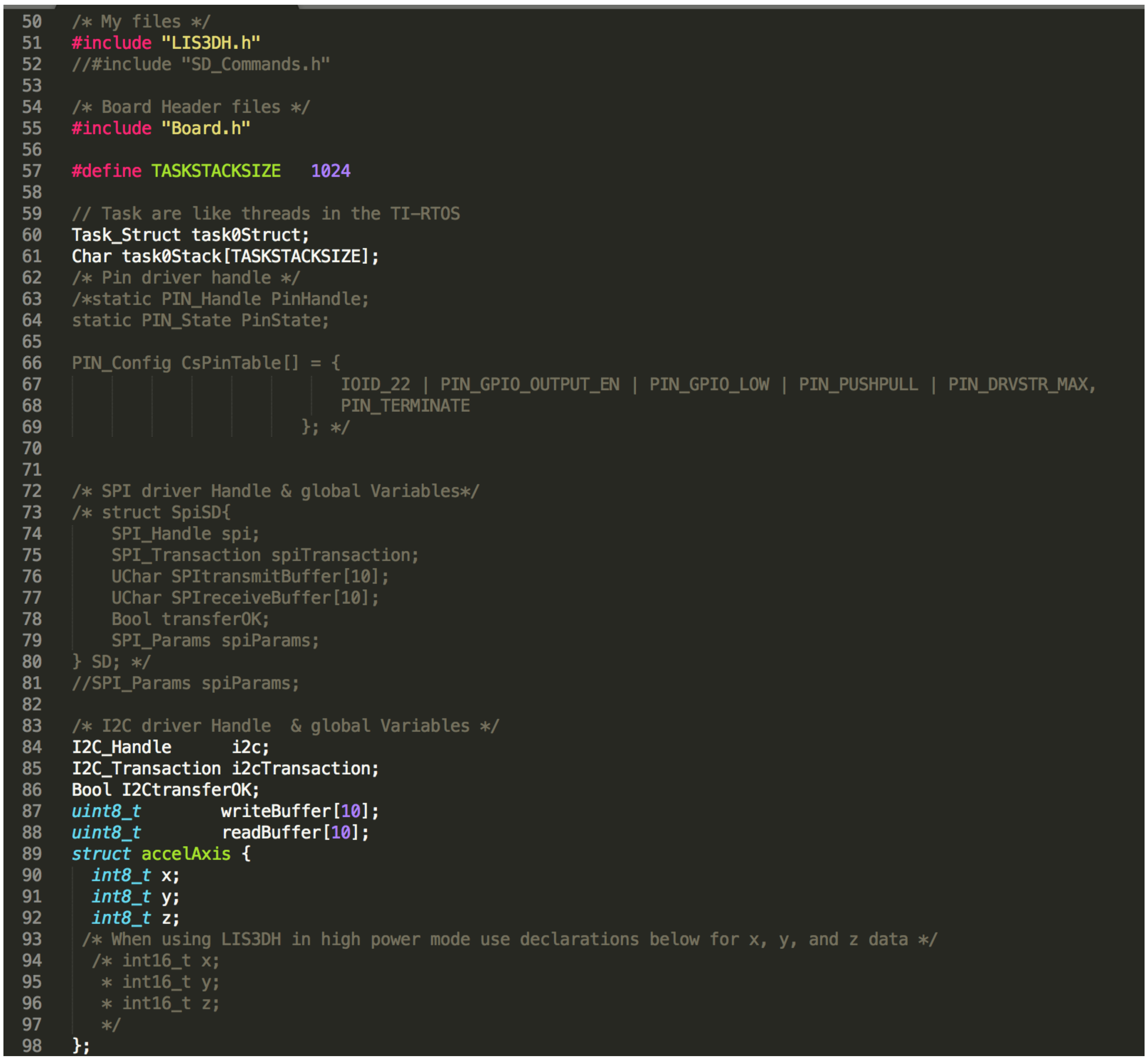
Figure 5. The project's critical path displayed as a PERT chart with optimistic and pessimistic time estimates for each task.

Appendix C

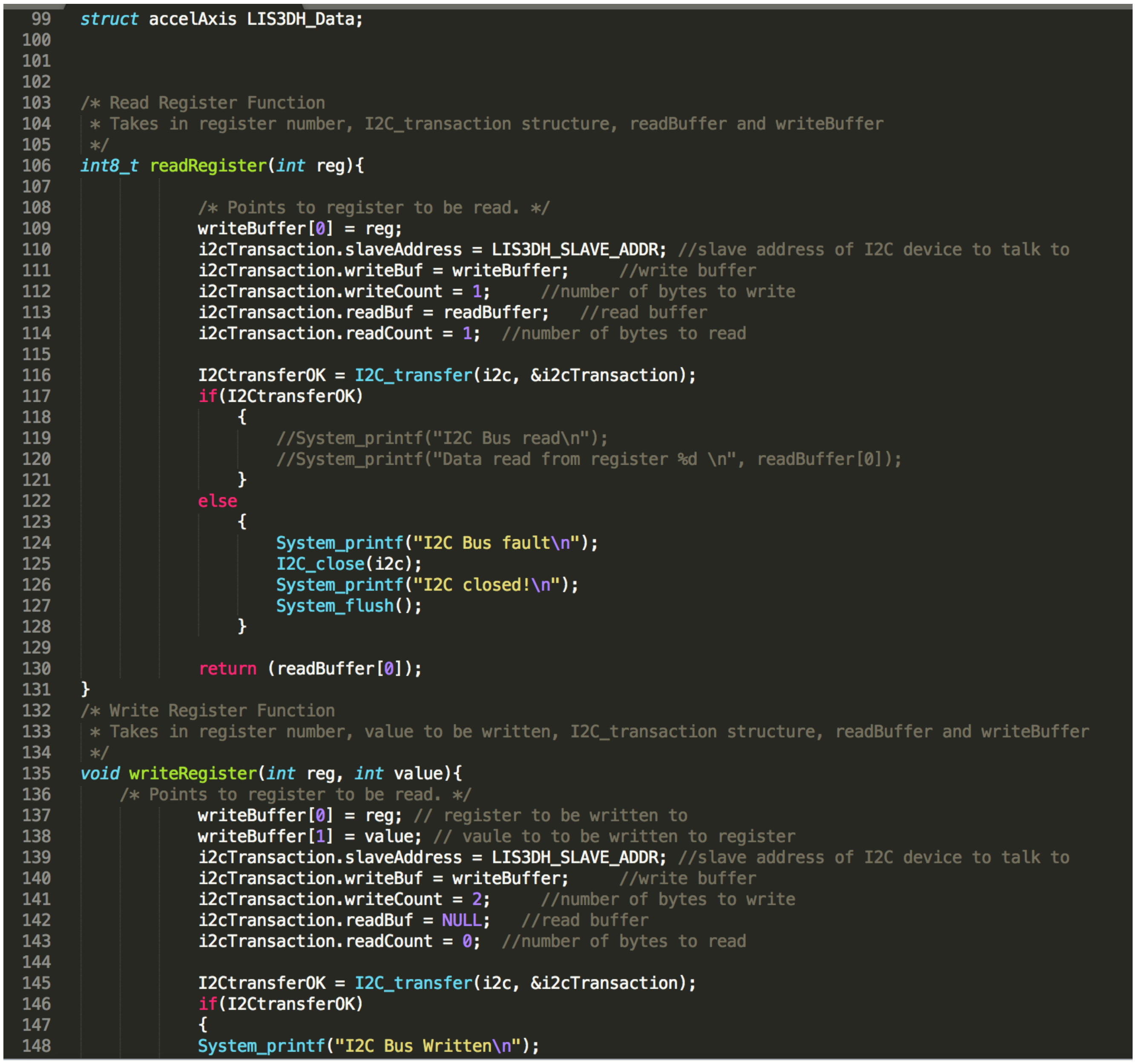
Screenshots of Main() code for the uC



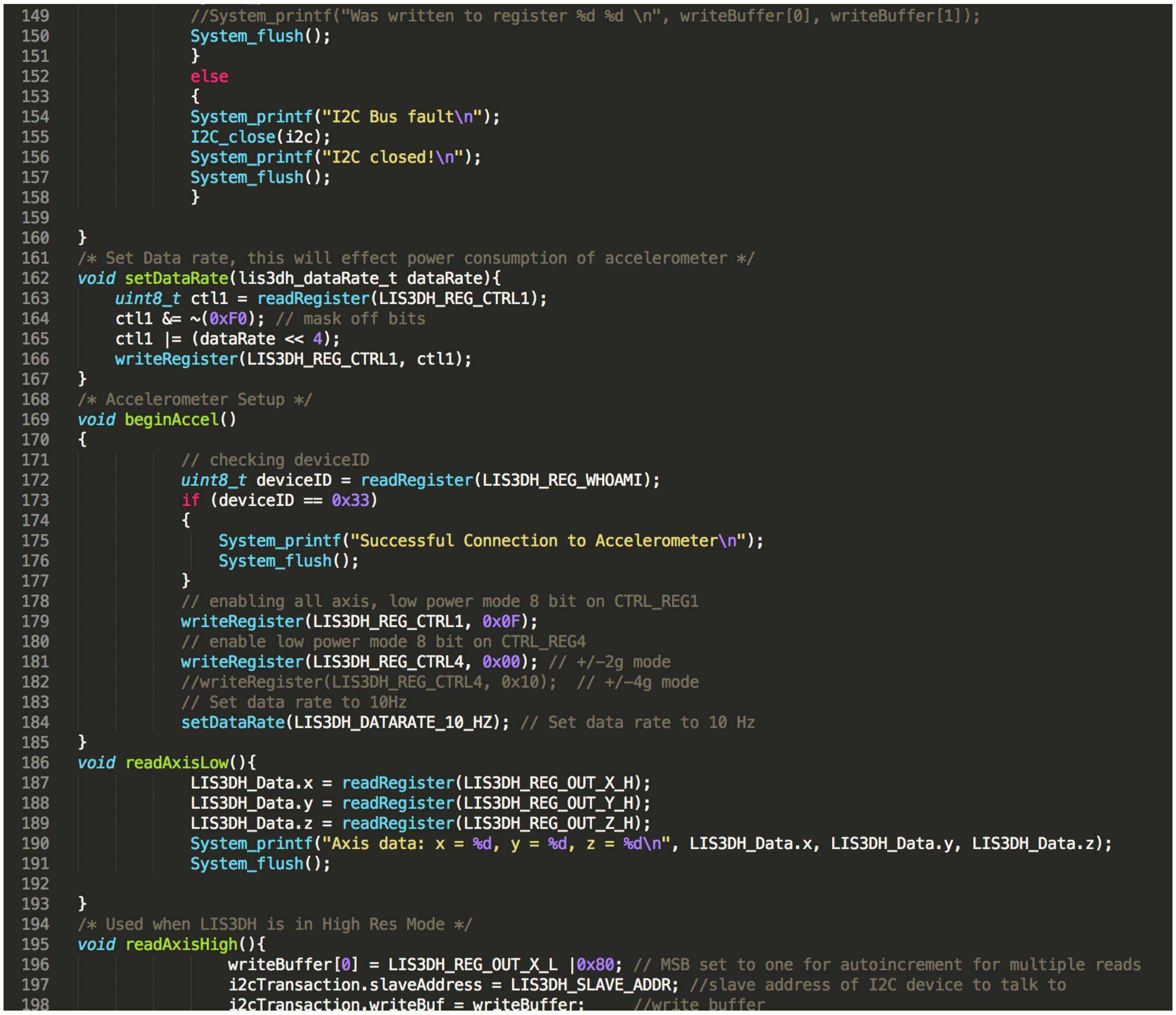
```
 * Copyright (c) 2015-2016, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/*
 *  ======== empty_min.c ========
 */
/* XDCtools Header files */
#include <xdc/std.h>
#include <xdc/runtime/System.h>

/* BIOS Header files */
#include <ti/sysbios/BIOS.h>
#include <ti/sysbios/knl/Task.h>
#include <ti/sysbios/knl/Clock.h>

/* TI-RTOS Header files */
#include <ti/drivers/I2C.h>
#include <ti/drivers/PIN.h>
#include <ti/drivers/SPI.h>

```



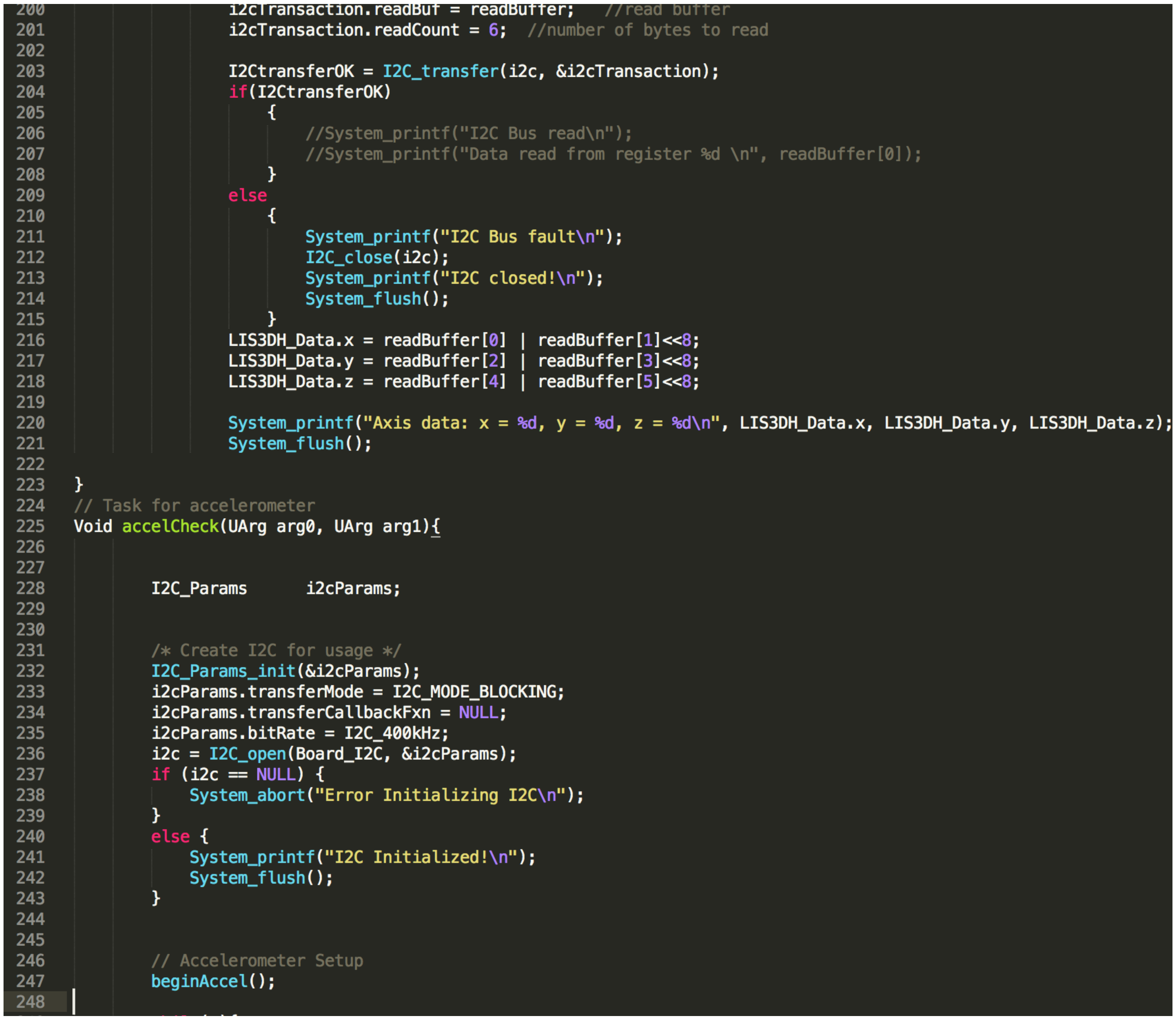
```c
/* My files */
#include "LIS3DH.h"
//#include "SD_Commands.h"

/* Board Header files */
#include "Board.h"

#define TASKSTACKSIZE   1024

// Task are like threads in the TI-RTOS
Task_Struct task0Struct;
Char task0Stack[TASKSTACKSIZE];
/* Pin driver handle */
/*static PIN_Handle PinHandle;
static PIN_State PinState;

PIN_Config CsPinTable[] = {
                           IOID_22 | PIN_GPIO_OUTPUT_EN | PIN_GPIO_LOW | PIN_PUSHPULL | PIN_DRVSTR_MAX,
                           PIN_TERMINATE
                       }; */


/* SPI driver Handle & global Variables*/
/* struct SpiSD{
    SPI_Handle spi;
    SPI_Transaction spiTransaction;
    UChar SPItransmitBuffer[10];
    UChar SPIreceiveBuffer[10];
    Bool transferOK;
    SPI_Params spiParams;
} SD; */
//SPI_Params spiParams;

/* I2C driver Handle  & global Variables */
I2C_Handle      i2c;
I2C_Transaction i2cTransaction;
Bool I2CtransferOK;
uint8_t        writeBuffer[10];
uint8_t        readBuffer[10];
struct accelAxis {
  int8_t x;
  int8_t y;
  int8_t z;
 /* When using LIS3DH in high power mode use declarations below for x, y, and z data */
  /* int16_t x;
   * int16_t y;
   * int16_t z;
   */
};
```



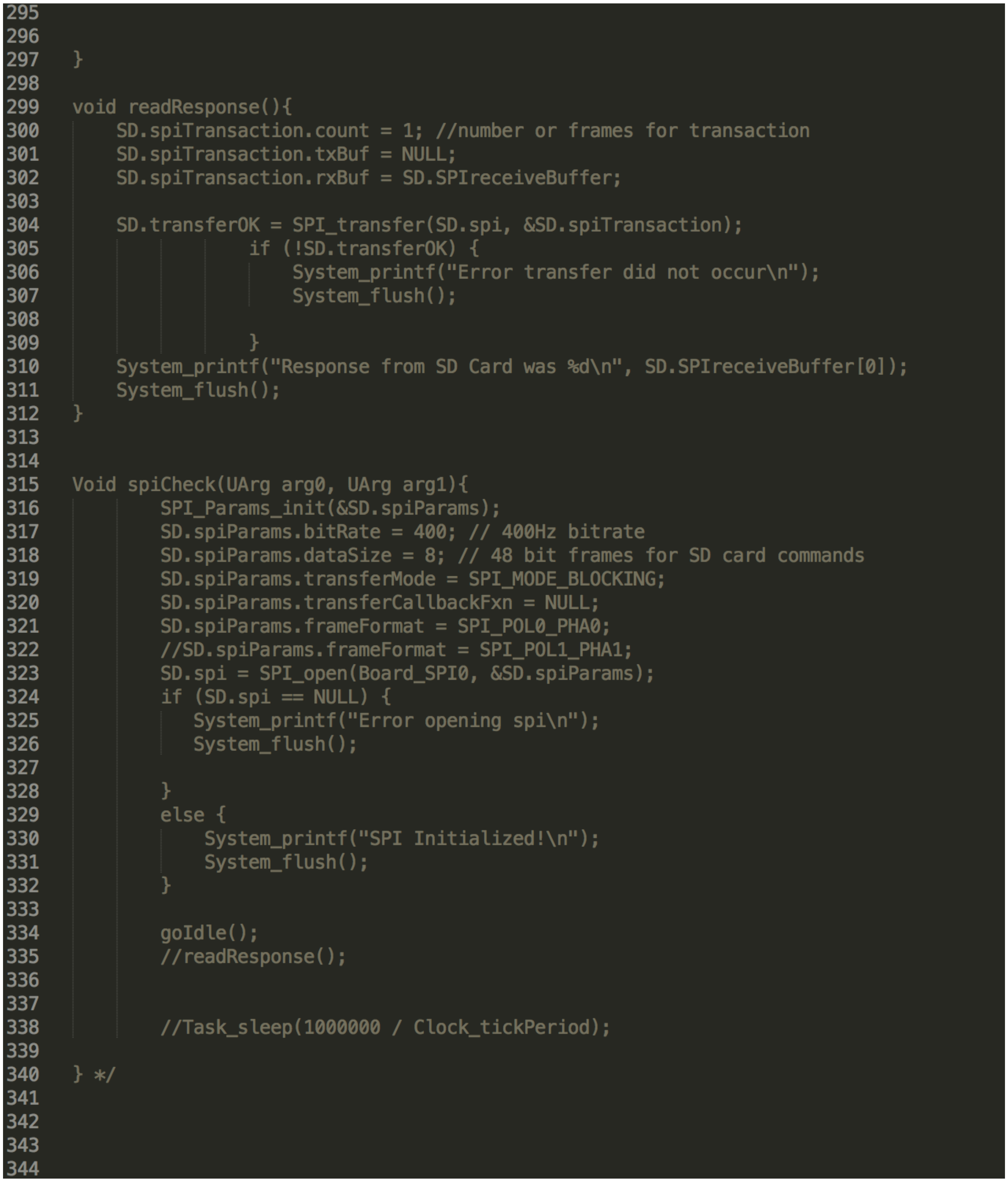
```c
struct accelAxis LIS3DH_Data;



/* Read Register Function
 * Takes in register number, I2C_transaction structure, readBuffer and writeBuffer
 */
int8_t readRegister(int reg){

            /* Points to register to be read. */
            writeBuffer[0] = reg;
            i2cTransaction.slaveAddress = LIS3DH_SLAVE_ADDR; //slave address of I2C device to talk to
            i2cTransaction.writeBuf = writeBuffer;     //write buffer
            i2cTransaction.writeCount = 1;      //number of bytes to write
            i2cTransaction.readBuf = readBuffer;    //read buffer
            i2cTransaction.readCount = 1;  //number of bytes to read

            I2CtransferOK = I2C_transfer(i2c, &i2cTransaction);
            if(I2CtransferOK)
                {
                    //System_printf("I2C Bus read\n");
                    //System_printf("Data read from register %d \n", readBuffer[0]);
                }
            else
                {
                    System_printf("I2C Bus fault\n");
                    I2C_close(i2c);
                    System_printf("I2C closed!\n");
                    System_flush();
                }

            return (readBuffer[0]);
}
/* Write Register Function
 * Takes in register number, value to be written, I2C_transaction structure, readBuffer and writeBuffer
 */
void writeRegister(int reg, int value){
    /* Points to register to be read. */
            writeBuffer[0] = reg; // register to be written to
            writeBuffer[1] = value; // vaule to to be written to register
            i2cTransaction.slaveAddress = LIS3DH_SLAVE_ADDR; //slave address of I2C device to talk to
            i2cTransaction.writeBuf = writeBuffer;     //write buffer
            i2cTransaction.writeCount = 2;      //number of bytes to write
            i2cTransaction.readBuf = NULL;    //read buffer
            i2cTransaction.readCount = 0;  //number of bytes to read

            I2CtransferOK = I2C_transfer(i2c, &i2cTransaction);
            if(I2CtransferOK)
            {
            System_printf("I2C Bus Written\n");
```



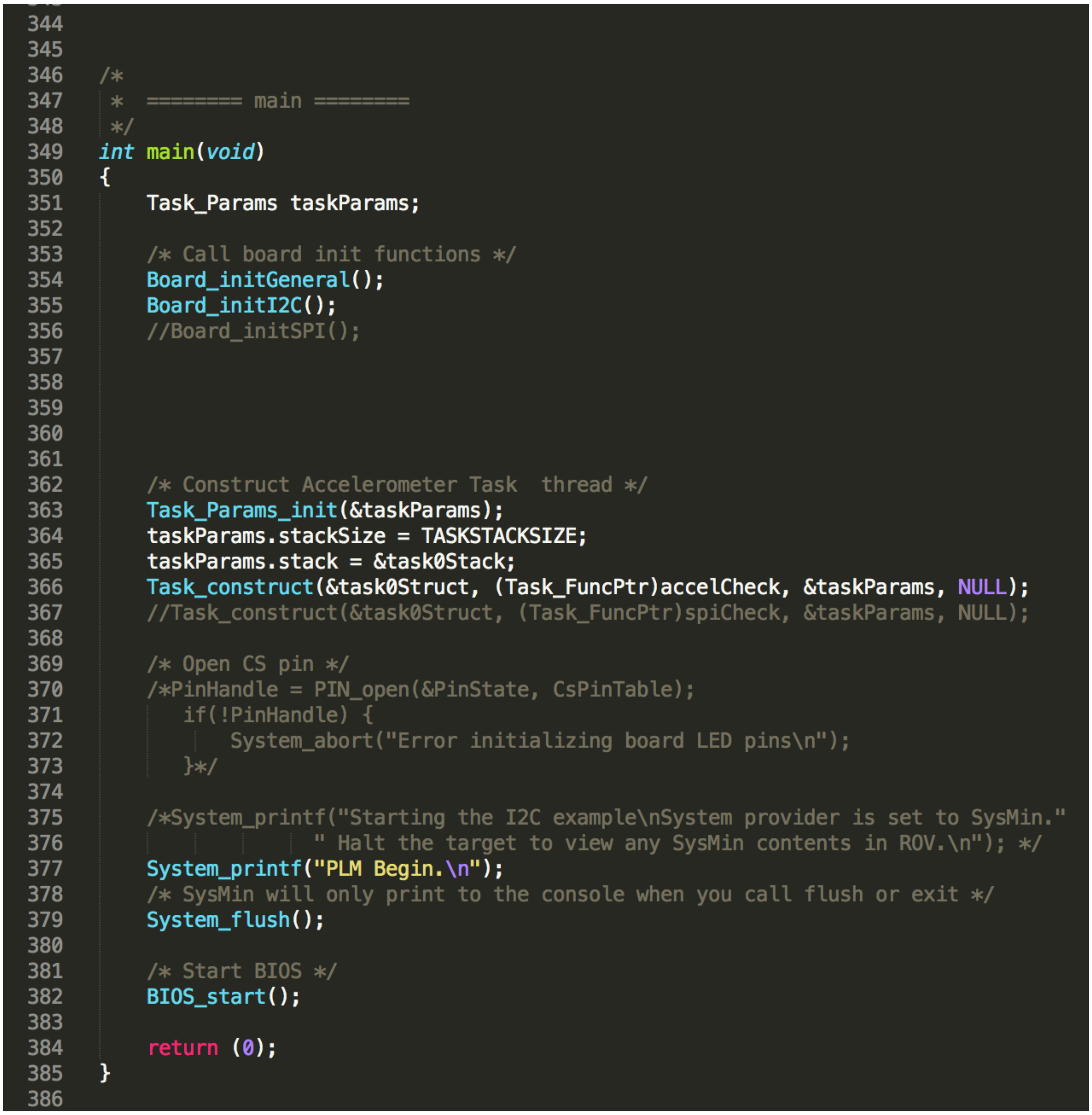
```
            //System_printf("Was written to register %d %d \n", writeBuffer[0], writeBuffer[1]);
            System_flush();
            }
            else
            {
            System_printf("I2C Bus fault\n");
            I2C_close(i2c);
            System_printf("I2C closed!\n");
            System_flush();
            }

}
/* Set Data rate, this will effect power consumption of accelerometer */
void setDataRate(lis3dh_dataRate_t dataRate){
    uint8_t ctl1 = readRegister(LIS3DH_REG_CTRL1);
    ctl1 &= ~(0xF0); // mask off bits
    ctl1 |= (dataRate << 4);
    writeRegister(LIS3DH_REG_CTRL1, ctl1);
}
/* Accelerometer Setup */
void beginAccel()
{
           // checking deviceID
           uint8_t deviceID = readRegister(LIS3DH_REG_WHOAMI);
           if (deviceID == 0x33)
           {
               System_printf("Successful Connection to Accelerometer\n");
               System_flush();
           }
           // enabling all axis, low power mode 8 bit on CTRL_REG1
           writeRegister(LIS3DH_REG_CTRL1, 0x0F);
           // enable low power mode 8 bit on CTRL_REG4
           writeRegister(LIS3DH_REG_CTRL4, 0x00); // +/-2g mode
           //writeRegister(LIS3DH_REG_CTRL4, 0x10);  // +/-4g mode
           // Set data rate to 10Hz
           setDataRate(LIS3DH_DATARATE_10_HZ); // Set data rate to 10 Hz
}
void readAxisLow(){
            LIS3DH_Data.x = readRegister(LIS3DH_REG_OUT_X_H);
            LIS3DH_Data.y = readRegister(LIS3DH_REG_OUT_Y_H);
            LIS3DH_Data.z = readRegister(LIS3DH_REG_OUT_Z_H);
            System_printf("Axis data: x = %d, y = %d, z = %d\n", LIS3DH_Data.x, LIS3DH_Data.y, LIS3DH_Data.z);
            System_flush();

}
/* Used when LIS3DH is in High Res Mode */
void readAxisHigh(){
                writeBuffer[0] = LIS3DH_REG_OUT_X_L |0x80; // MSB set to one for autoincrement for multiple reads
                i2cTransaction.slaveAddress = LIS3DH_SLAVE_ADDR; //slave address of I2C device to talk to
                i2cTransaction.writeBuf = writeBuffer;     //write buffer
```



```
                i2cTransaction.readBuf = readBuffer;    //read buffer
                i2cTransaction.readCount = 6;  //number of bytes to read

                I2CtransferOK = I2C_transfer(i2c, &i2cTransaction);
                if(I2CtransferOK)
                    {
                        //System_printf("I2C Bus read\n");
                        //System_printf("Data read from register %d \n", readBuffer[0]);
                    }
                else
                    {
                        System_printf("I2C Bus fault\n");
                        I2C_close(i2c);
                        System_printf("I2C closed!\n");
                        System_flush();
                    }
                LIS3DH_Data.x = readBuffer[0] | readBuffer[1]<<8;
                LIS3DH_Data.y = readBuffer[2] | readBuffer[3]<<8;
                LIS3DH_Data.z = readBuffer[4] | readBuffer[5]<<8;

                System_printf("Axis data: x = %d, y = %d, z = %d\n", LIS3DH_Data.x, LIS3DH_Data.y, LIS3DH_Data.z);
                System_flush();

}
// Task for accelerometer
Void accelCheck(UArg arg0, UArg arg1){


        I2C_Params      i2cParams;


        /* Create I2C for usage */
        I2C_Params_init(&i2cParams);
        i2cParams.transferMode = I2C_MODE_BLOCKING;
        i2cParams.transferCallbackFxn = NULL;
        i2cParams.bitRate = I2C_400kHz;
        i2c = I2C_open(Board_I2C, &i2cParams);
        if (i2c == NULL) {
            System_abort("Error Initializing I2C\n");
        }
        else {
            System_printf("I2C Initialized!\n");
            System_flush();
        }


        // Accelerometer Setup
        beginAccel();

```



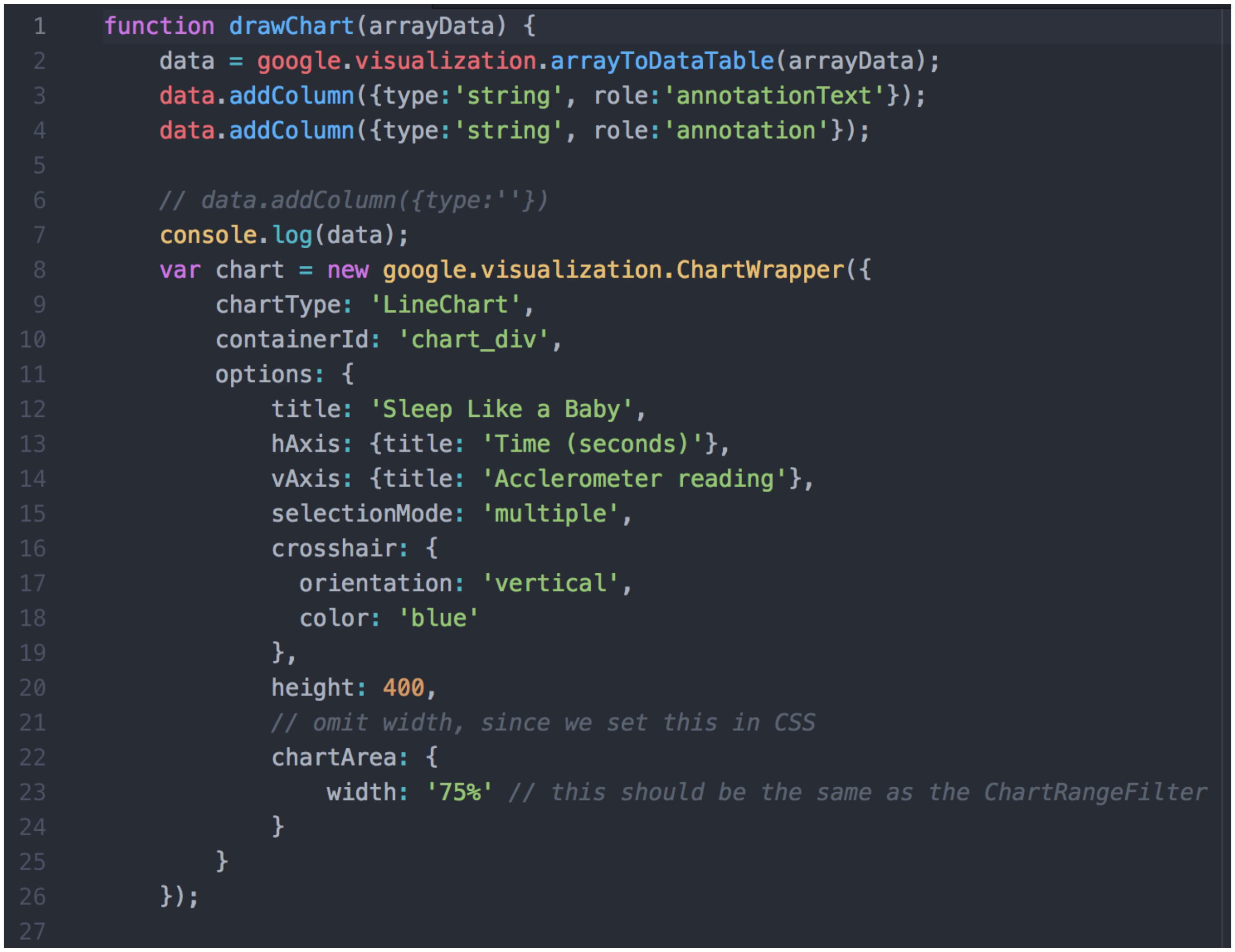
```


}

void readResponse(){
    SD.spiTransaction.count = 1; //number or frames for transaction
    SD.spiTransaction.txBuf = NULL;
    SD.spiTransaction.rxBuf = SD.SPIreceiveBuffer;

    SD.transferOK = SPI_transfer(SD.spi, &SD.spiTransaction);
                if (!SD.transferOK) {
                    System_printf("Error transfer did not occur\n");
                    System_flush();

                }
    System_printf("Response from SD Card was %d\n", SD.SPIreceiveBuffer[0]);
    System_flush();
}


Void spiCheck(UArg arg0, UArg arg1){
        SPI_Params_init(&SD.spiParams);
        SD.spiParams.bitRate = 400; // 400Hz bitrate
        SD.spiParams.dataSize = 8; // 48 bit frames for SD card commands
        SD.spiParams.transferMode = SPI_MODE_BLOCKING;
        SD.spiParams.transferCallbackFxn = NULL;
        SD.spiParams.frameFormat = SPI_POL0_PHA0;
        //SD.spiParams.frameFormat = SPI_POL1_PHA1;
        SD.spi = SPI_open(Board_SPI0, &SD.spiParams);
        if (SD.spi == NULL) {
           System_printf("Error opening spi\n");
           System_flush();

        }
        else {
            System_printf("SPI Initialized!\n");
            System_flush();
        }

        goIdle();
        //readResponse();


        //Task_sleep(1000000 / Clock_tickPeriod);

} */
```



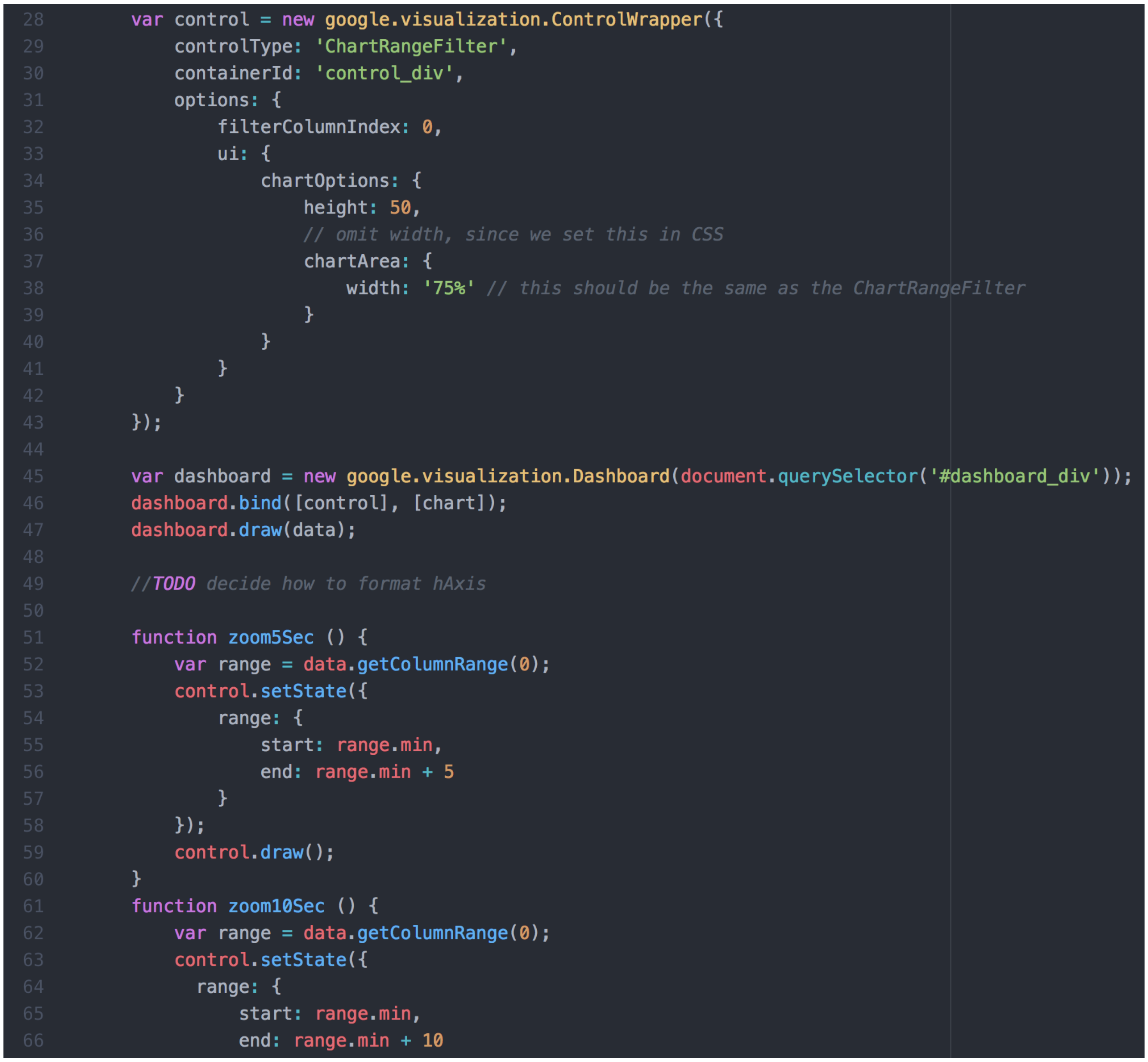
```


/*
 * ======== main ========
 */
int main(void)
{
    Task_Params taskParams;

    /* Call board init functions */
    Board_initGeneral();
    Board_initI2C();
    //Board_initSPI();





    /* Construct Accelerometer Task  thread */
    Task_Params_init(&taskParams);
    taskParams.stackSize = TASKSTACKSIZE;
    taskParams.stack = &task0Stack;
    Task_construct(&task0Struct, (Task_FuncPtr)accelCheck, &taskParams, NULL);
    //Task_construct(&task0Struct, (Task_FuncPtr)spiCheck, &taskParams, NULL);

    /* Open CS pin */
    /*PinHandle = PIN_open(&PinState, CsPinTable);
       if(!PinHandle) {
           System_abort("Error initializing board LED pins\n");
       }*/

    /*System_printf("Starting the I2C example\nSystem provider is set to SysMin."
                  " Halt the target to view any SysMin contents in ROV.\n"); */
    System_printf("PLM Begin.\n");
    /* SysMin will only print to the console when you call flush or exit */
    System_flush();

    /* Start BIOS */
    BIOS_start();

    return (0);
}

```

Appendix D

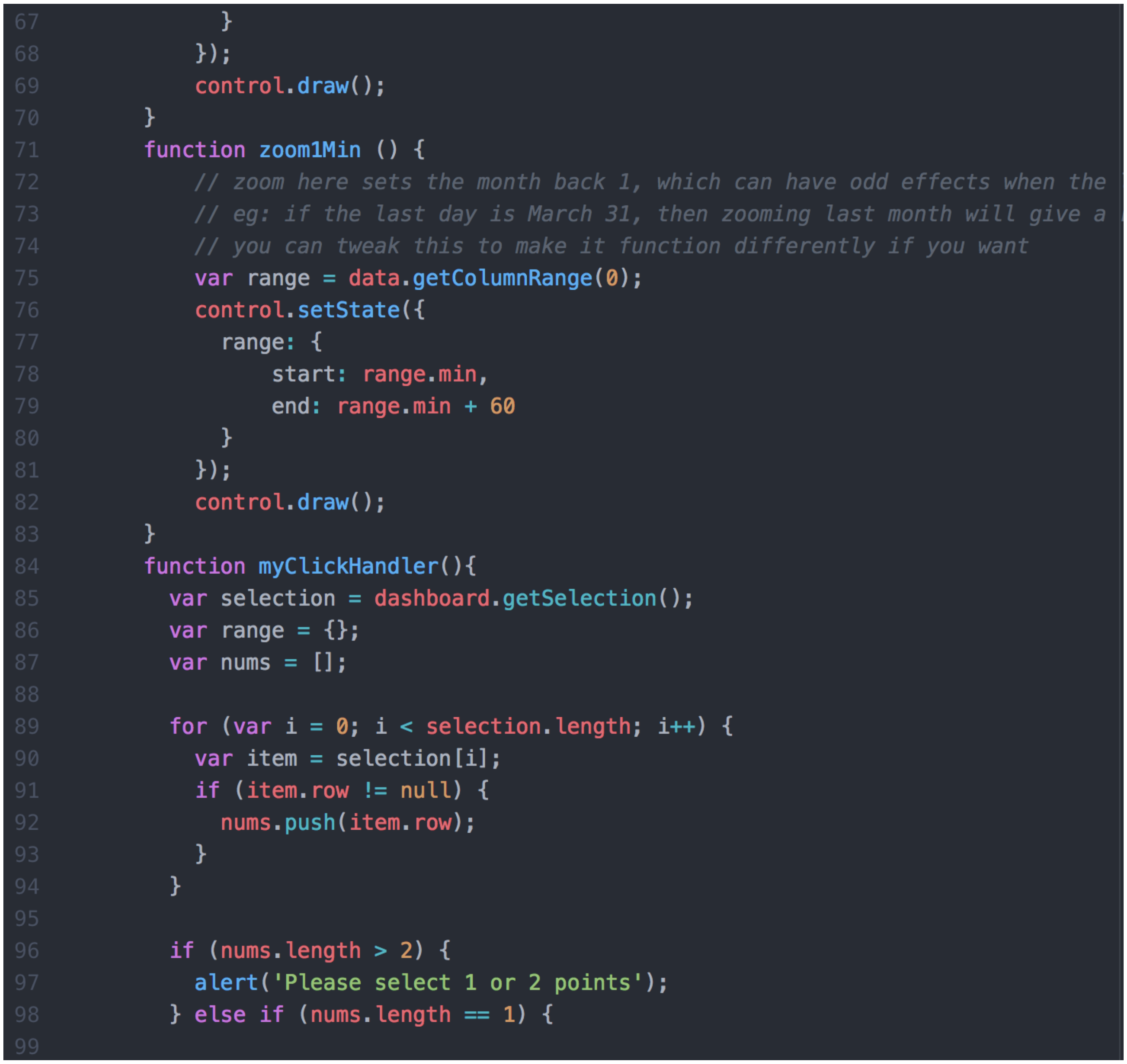
Screenshots of main Javascript file used to render the GUI



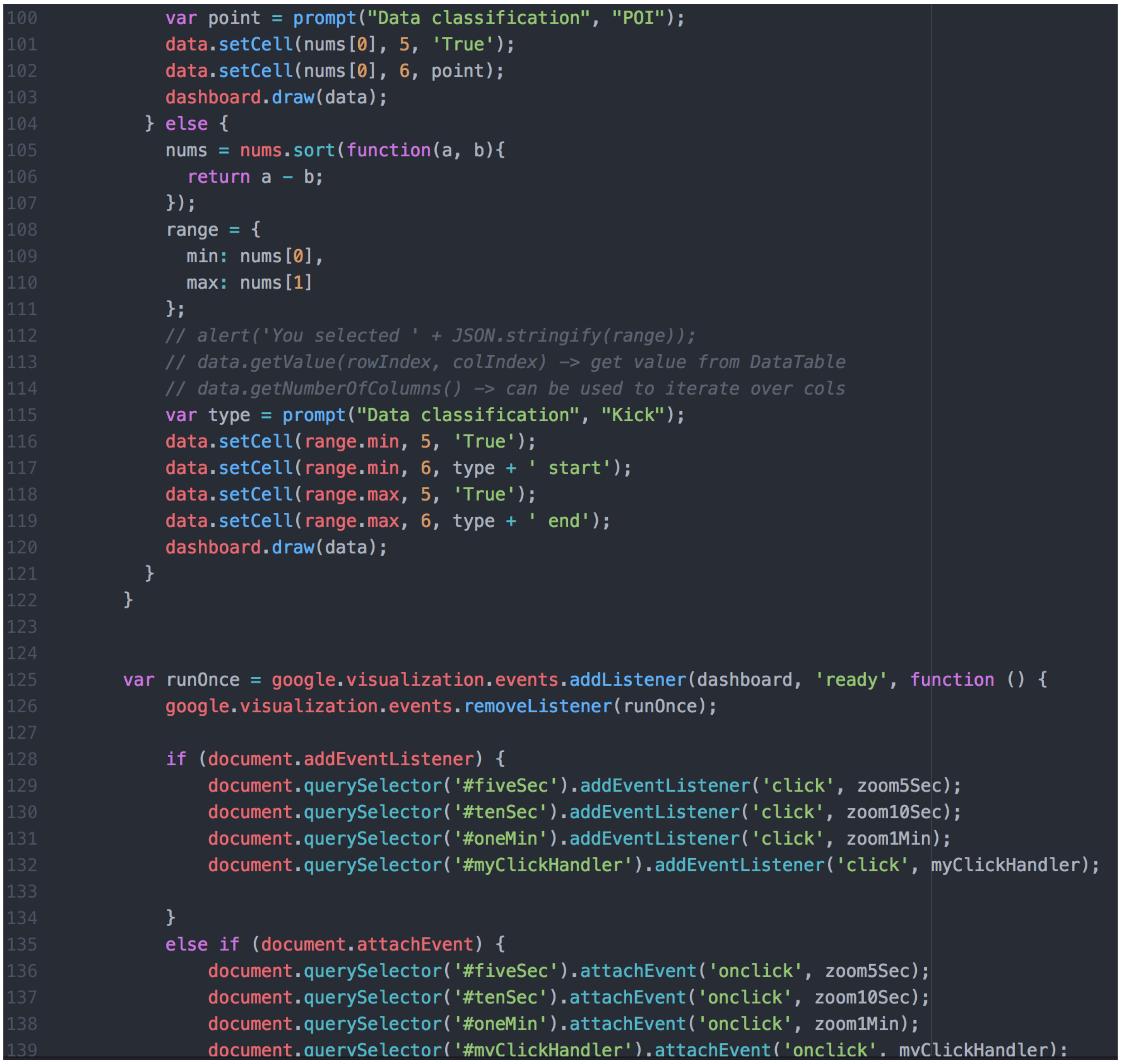
```
function drawChart(arrayData) {
    data = google.visualization.arrayToDataTable(arrayData);
    data.addColumn({type:'string', role:'annotationText'});
    data.addColumn({type:'string', role:'annotation'});

    // data.addColumn({type:''})
    console.log(data);
    var chart = new google.visualization.ChartWrapper({
        chartType: 'LineChart',
        containerId: 'chart_div',
        options: {
            title: 'Sleep Like a Baby',
            hAxis: {title: 'Time (seconds)'},
            vAxis: {title: 'Acclerometer reading'},
            selectionMode: 'multiple',
            crosshair: {
              orientation: 'vertical',
              color: 'blue'
            },
            height: 400,
            // omit width, since we set this in CSS
            chartArea: {
                width: '75%' // this should be the same as the ChartRangeFilter
            }
        }
    });

```



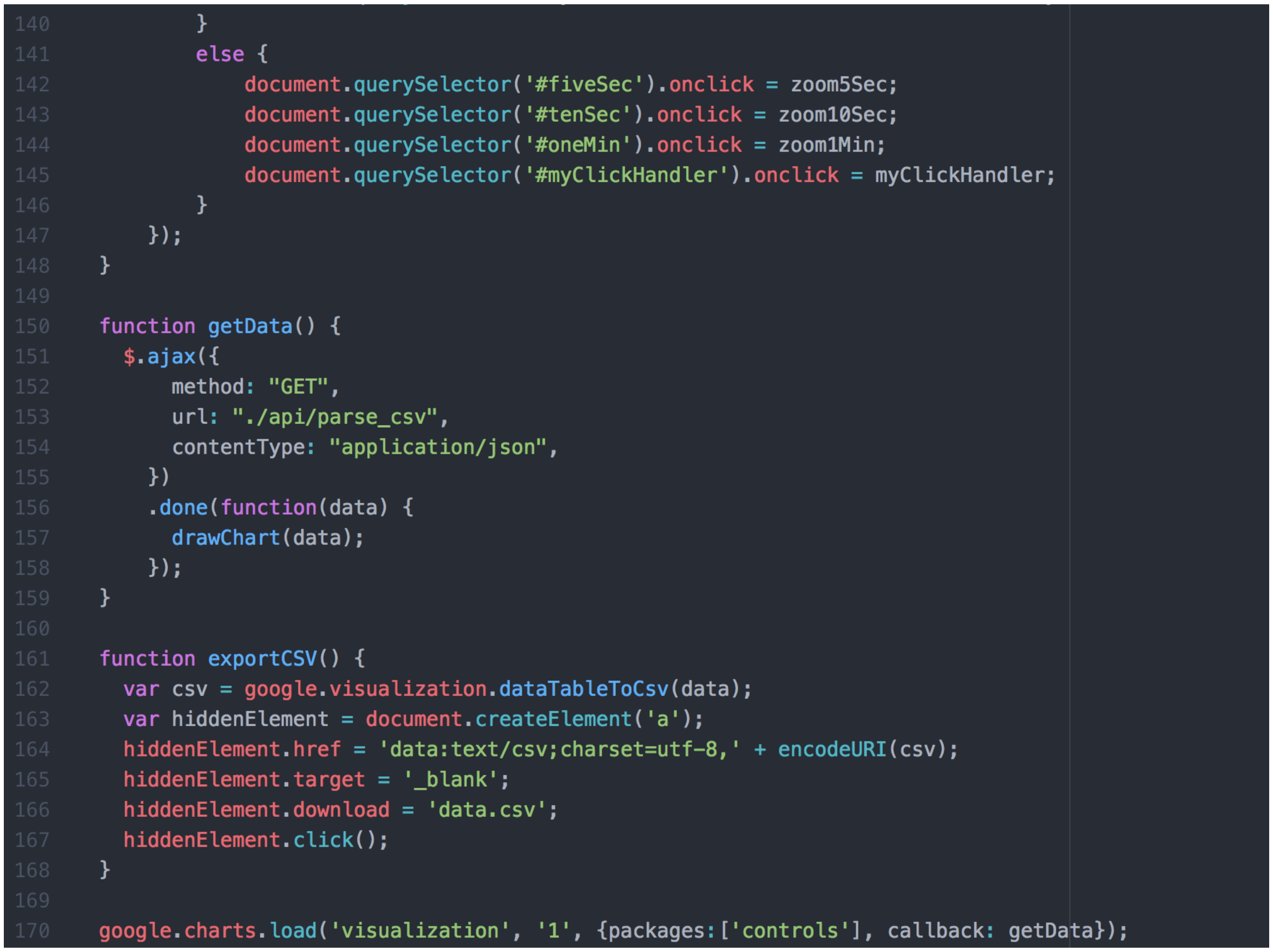
```
        var control = new google.visualization.ControlWrapper({
            controlType: 'ChartRangeFilter',
            containerId: 'control_div',
            options: {
                filterColumnIndex: 0,
                ui: {
                    chartOptions: {
                        height: 50,
                        // omit width, since we set this in CSS
                        chartArea: {
                            width: '75%' // this should be the same as the ChartRangeFilter
                        }
                    }
                }
            }
        });

        var dashboard = new google.visualization.Dashboard(document.querySelector('#dashboard_div'));
        dashboard.bind([control], [chart]);
        dashboard.draw(data);

        //TODO decide how to format hAxis

        function zoom5Sec () {
            var range = data.getColumnRange(0);
            control.setState({
                range: {
                    start: range.min,
                    end: range.min + 5
                }
            });
            control.draw();
        }
        function zoom10Sec () {
            var range = data.getColumnRange(0);
            control.setState({
              range: {
                  start: range.min,
                  end: range.min + 10
```



```
              }
            });
            control.draw();
        }
        function zoom1Min () {
            // zoom here sets the month back 1, which can have odd effects when the
            // eg: if the last day is March 31, then zooming last month will give a
            // you can tweak this to make it function differently if you want
            var range = data.getColumnRange(0);
            control.setState({
              range: {
                  start: range.min,
                  end: range.min + 60
              }
            });
            control.draw();
        }
        function myClickHandler(){
          var selection = dashboard.getSelection();
          var range = {};
          var nums = [];

          for (var i = 0; i < selection.length; i++) {
            var item = selection[i];
            if (item.row != null) {
              nums.push(item.row);
            }
          }

          if (nums.length > 2) {
            alert('Please select 1 or 2 points');
          } else if (nums.length == 1) {

```



```
        var point = prompt("Data classification", "POI");
        data.setCell(nums[0], 5, 'True');
        data.setCell(nums[0], 6, point);
        dashboard.draw(data);
      } else {
        nums = nums.sort(function(a, b){
          return a - b;
        });
        range = {
          min: nums[0],
          max: nums[1]
        };
        // alert('You selected ' + JSON.stringify(range));
        // data.getValue(rowIndex, colIndex) -> get value from DataTable
        // data.getNumberOfColumns() -> can be used to iterate over cols
        var type = prompt("Data classification", "Kick");
        data.setCell(range.min, 5, 'True');
        data.setCell(range.min, 6, type + ' start');
        data.setCell(range.max, 5, 'True');
        data.setCell(range.max, 6, type + ' end');
        dashboard.draw(data);
      }
    }


    var runOnce = google.visualization.events.addListener(dashboard, 'ready', function () {
        google.visualization.events.removeListener(runOnce);

        if (document.addEventListener) {
            document.querySelector('#fiveSec').addEventListener('click', zoom5Sec);
            document.querySelector('#tenSec').addEventListener('click', zoom10Sec);
            document.querySelector('#oneMin').addEventListener('click', zoom1Min);
            document.querySelector('#myClickHandler').addEventListener('click', myClickHandler);

        }
        else if (document.attachEvent) {
            document.querySelector('#fiveSec').attachEvent('onclick', zoom5Sec);
            document.querySelector('#tenSec').attachEvent('onclick', zoom10Sec);
            document.querySelector('#oneMin').attachEvent('onclick', zoom1Min);
            document.querySelector('#myClickHandler').attachEvent('onclick', myClickHandler);
```



```
        }
        else {
            document.querySelector('#fiveSec').onclick = zoom5Sec;
            document.querySelector('#tenSec').onclick = zoom10Sec;
            document.querySelector('#oneMin').onclick = zoom1Min;
            document.querySelector('#myClickHandler').onclick = myClickHandler;
        }
    });
}

function getData() {
  $.ajax({
      method: "GET",
      url: "./api/parse_csv",
      contentType: "application/json",
    })
    .done(function(data) {
      drawChart(data);
    });
}

function exportCSV() {
  var csv = google.visualization.dataTableToCsv(data);
  var hiddenElement = document.createElement('a');
  hiddenElement.href = 'data:text/csv;charset=utf-8,' + encodeURI(csv);
  hiddenElement.target = '_blank';
  hiddenElement.download = 'data.csv';
  hiddenElement.click();
}

google.charts.load('visualization', '1', {packages:['controls'], callback: getData});
```