Bridge Inspection Robot

ECE4012 Senior Design Project

Section L01, Bridge Inspection Robot Team

Project Advisor, Dr. Patricio A. Vela

Erikzzon S. Latoja, elatoja@gatech.edu, Team Leader

Justin Tamayo, jtamayo6@gatech.edu, Web Master

Sean Csukas, sean.csukas@gatech.edu

Kristen Fernandez, kfernandez7@gatech.edu

Sanmeshkumar Udhayakumar, sudhayakumar3@gatech.edu

Submitted

3 May 2017

Executive Summary

The maintenance of civil infrastructure systems is a constant challenge faced by today's engineers. Structural health monitoring is critical for identifying sections of civil infrastructure in need of repair, but traditional methods for doing so are time-consuming and costly. As an alternate solution, a network of wirelessly sensing robots can be developed to overcome these obstacles.

The main objective of the team was to design a wireless sensing robot that can be used for structural health monitoring of civil structures, particularly bridges. The Bridge Inspection Robot consisted of a computing core, various sensors, a wireless transceiver, and a custom movement mechanism that will permit the robot to autonomously scale bridges. The old wheeled design of the robot was limited in its ability to move along paths that were not a straight line. The team proposed several new designs that will allow the robot to turn corners and take non-linear paths more efficiently than the original design and settled on a three-wheeled robot with two powered wheels in the front and an omni-wheel in the back. New electrical and mechanical components will also be worked into the design to ensure that the robot has the most up-to-date parts.

On a larger scale, this Bridge Inspection Robot would be part of a network of wireless sensing robots. This project focused on just prototyping a single robot as a proof of concept that the larger network of robots is feasible. The total cost of one Bridge Inspection Robot is estimated to be $498.

Table of Contents

Bridge Inspection Robot

1 Introduction

1.1 Objective

The main objective of the team was to design a wireless sensing robot that can be used for structural health monitoring of civil structures, particularly bridges. This Bridge Inspection Robot would be capable of scaling structures and wirelessly sending accelerometer data to an external server for processing. The robot would be constructed to be as lightweight as possible to allow it to be carried by a quadcopter for placement onto a bridge. The current design of the robot was reworked from the ground-up from the original robot designed in 2011 to create a prototype that has increased maneuverability and improved mechanical and electrical components. Figure 1 shows the old wheeled prototype of the robot that was created in 2011. This design, although functional, was limited in its ability to move along paths that were not a straight line. The team proposed several new designs that allowed the robot to turn corners and take non-linear paths more efficiently than the original design and settled on a three-wheeled robot with two powered wheels in the front and an omni-wheel in the back. New electrical and mechanical components were also worked into the design to ensure that the robot has the most up-to-date parts.

On a larger scale, this Bridge Inspection Robot would be part of a network of wireless sensing robots. These robots would take measurements in one small neighborhood and then move on to the next part of the bridge, until the whole structure is scanned. However, this project focused on just prototyping a single robot as a proof of concept that the larger network of robots is feasible.

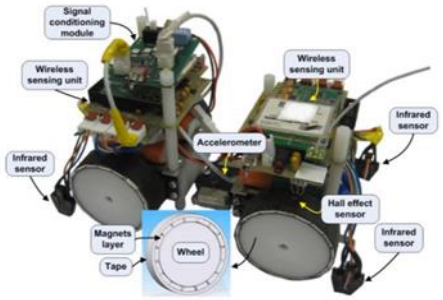


Figure 1. Wheeled prototype of the robot created in 2011

1.2 Motivation

The repair and maintenance of civil infrastructure systems is a constant challenge faced by today's engineers. Visual inspections of bridges have been shown to be highly subjective, as different inspectors can give drastically different condition ratings for the same bridge. Conducting a visual inspection also only shows damage that is visible at the surface, leaving damage that is below the surface undetected [1]. As an alternative solution, structural health monitoring (SHM) systems are widely used to assess the conditions of civil structures. In a SHM system, accelerometers and other types of sensors collect data and monitor structural behavior [2]. Traditionally, cables and wires connect sensors to a central server, but these systems are typically high cost and time-consuming to install [3]. The development of a network of wirelessly sensing robots for SHM overcomes these difficulties [4]. The team aspired to create a new design of the Bridge Inspection Robot that is not only functional, but also performs its functions with improved maneuverability.

1.3 Background

There has been an increase in research for developing small-scale agile robots for inspecting engineered structures. These robots are typically used individually, and not in a mobile sensing network to provide measurements at multiple locations [4]. These robots also employ multiple methods to navigate different kinds of surfaces. For example, a robot with two magnetic wheels in a motorbike arrangement was developed to inspect the inner casing of complex-shaped metal pipes [5]. One kind of wall-climbing robot was developed using elastomer dry adhesion [6]; another uses claw-gripping to climb walls [7].

Recently, a model helicopter was developed to serve as a mobile host for charging and communicating with wireless sensors [8]. However, there are currently no products on the market that can dynamically move about a structure for the purposes of structural health monitoring.

2 Project Description and Goals

The team designed and built a prototype robot that can navigate bridges and record structural vibration data, which can then be used to monitor the structural health of these bridges over time. Components included a 32-bit microcontroller, a high-resolution accelerometer, IR sensors, a gyroscope sensor, a GPS sensor, and a wireless transceiver. The robot's movement was implemented through a permanent-magnet three-wheeled design, which allowed the robot to safely move along steel bridges in two dimensions. A PC wirelessly connected to the robot to send commands and store received sensor data. Features for the robot include:

- Ability to horizontally and vertically traverse steel bridges
- Measure bridge vibrations at low frequencies
- Wirelessly transmit vibration data to a PC
- Drone (quadcopter) deployment and retrieval
- $498 approximate build cost

3 Technical Specifications & Verification

TABLE I

TECHNICAL SPECIFICATIONS OF THE BRIDGE INSPECTION ROBOT

Characteristic	Specification
Magnet Holding Force	Robot holds position and orientation vertically and inverted
Operational Lifetime	Battery voltage drops 0.25 V from full charge over 1 hour
Accelerometer Range and Accuracy	0-50 Hz ± 0.5 Hz
Robot Size	0.24 m x 0.14 m x 0.20 m
Weight	Total Robot Mass 1.026 kg
Wireless Communication Distance	Robot sends and receives data at least 800 m away from base PC
Avoiding Falls	IR Sensors implemented to detect edges

4 Design Approach and Details

4.1 Design Approach

The Bridge Inspection Robot utilized a permanent-magnet three-wheeled design that permitted it to navigate bridges. Other critical components include a microcontroller, various sensors, and a wireless transceiver.

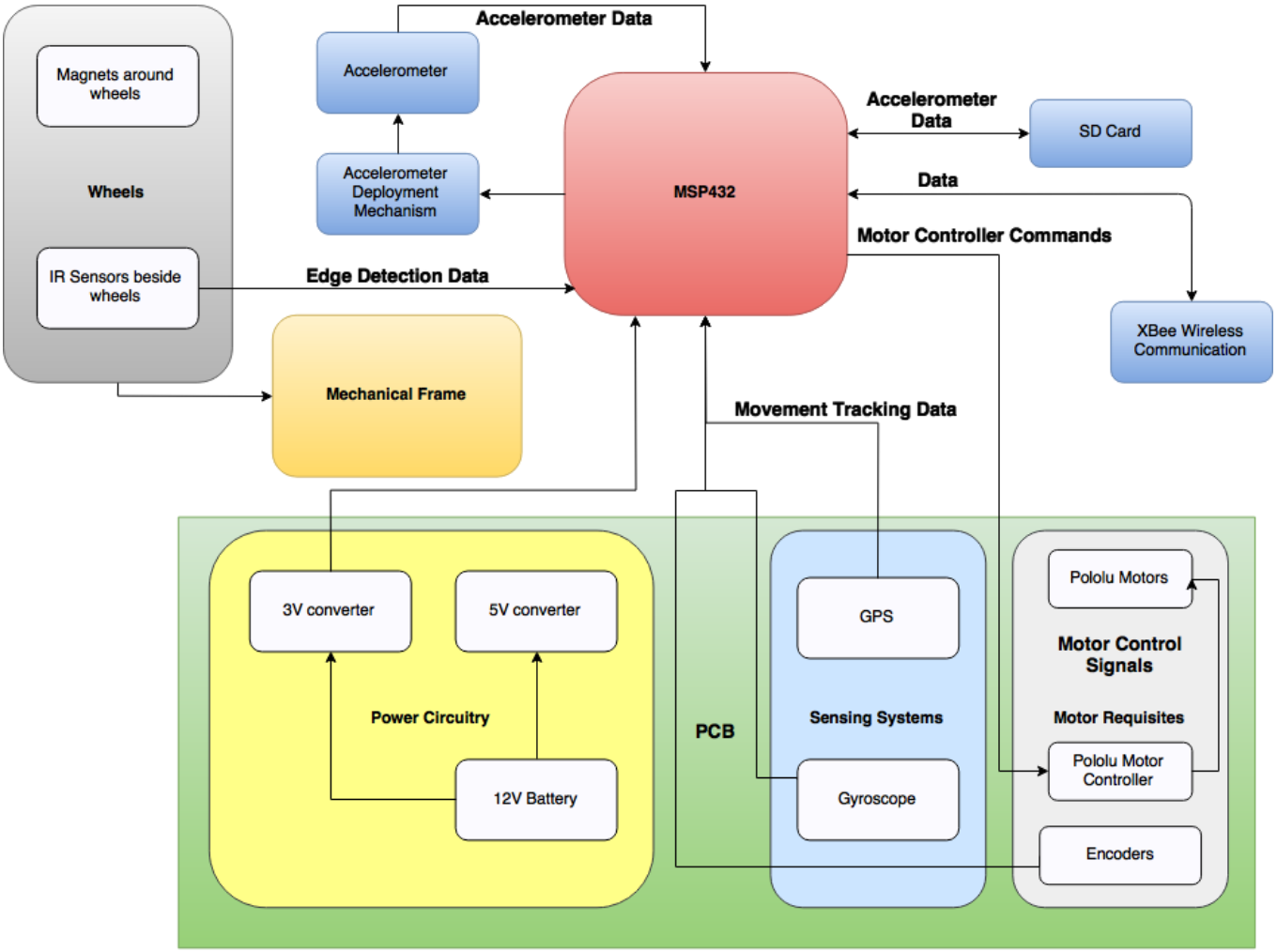


Figure 2. Block Diagram of Robot Operation

4.1.1 Robot Mobility

4.1.1.1 Three-Wheeled Design

The solution to the movement problem posed by the previous design is to replace the two static back wheels with a single omni wheel. This design as shown in Figure 7 features a single rigid body with two motorized wheels at the front and an omni wheel in the back. Permanent magnets surround the two motorized wheels are placed adjacent to the back wheel. This design would retain all the positive characteristics of the original flexure-based design.

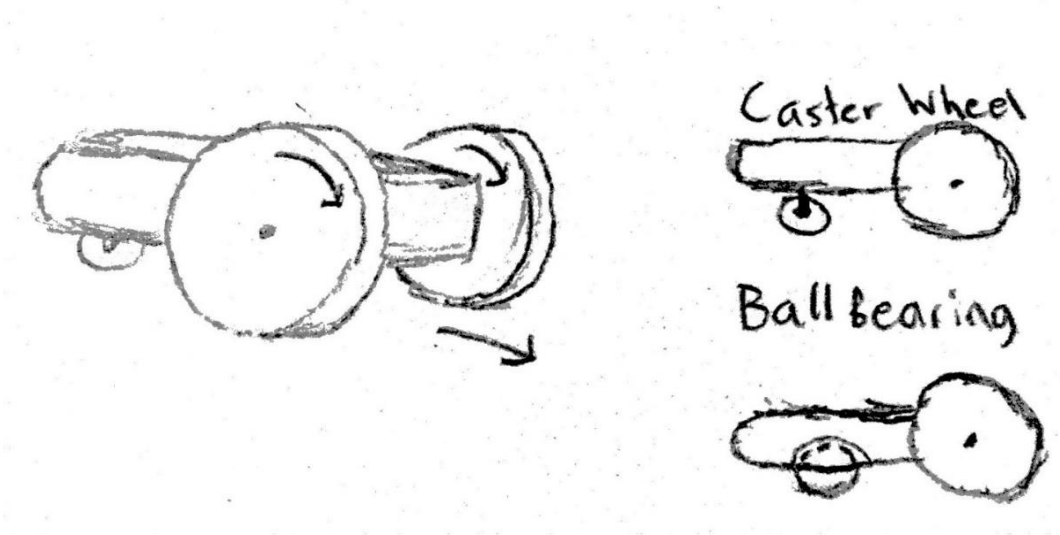


Figure 3. The design for a three-wheeled robot, with either a caster wheel or ball bearing in the back

To ensure that this design could traverse corners, the team carefully designed the robot so that the rigid body is raised along the structure between the two wheels, such that it will not hit or scrape the surface while it is rounding a corner of up to 90 degrees.

To ensure that this design could traverse corners, the team would also need carefully design the robot so that the rigid body is raised around the two front wheels, so that it will not hit or scrape the surface while it is rounding a corner.

4.1.1.2 Mobility Choice

The team decided upon the three-wheeled design as the choice for the robot's mobility. Not only does this robot overcome the forward movements that the two-wheeled robot encountered, but it retains all the pros of that design as well. Also, this design trumped the four-wheeled robot in terms of its mobility and lower weight and power consumption.

4.1.2 Motor Selection

In the chosen three-wheeled design, the front two wheels are each driven by one motor. The third is passive. The torque requirements can be seen in Eq. 1. This is based on the maximum torque scenario of driving vertically upward. A weight of 1kg and wheel radius of 5 cm are given. A maximum acceleration of 5m/s is assumed.

$$\tau_{max} = m * a * r = 1kg * \left(10\frac{m}{s^2} + 5\frac{m}{s^2}\right) * 0.05m = 0.75\ N \cdot m = 7.64\ \text{kg} \cdot \text{cm} \qquad (1)$$

Given two motors, 3.82 kg-cm per motor is the minimum torque for each motor. Both continuous servos and brushless DC motors were considered.

The Pololu low power 12V metal gearmotor with 99:1 gear reduction, as shown in Figure 4, provides an approximate maximum torque of 8.2 kg-cm at 55rpm, which is sufficient to provide the required torque without reaching the recommended continuous limit of the motor, which is 25% less than the maximum. Using the accompanying 90mm x 10mm wheels also provided by Pololu, at max RPM, the robot would travel at 0.1m/s, which is well above the desired speed. The motor also includes an encoder, which will be used in motion control and tracking.

Figure 4. Pololu low power 12V metal gearmotor with 99:1 gear reduction and encoder

4.1.3 Motion Control and Tracking

For the scope of the project in the time available for the initial design, two assumptions will be made. First, it will be assumed the initial orientation of the robot will be known, and thus, localization will not be incorporated in this design. In a practical application, one can imagine the Bridge Inspection Robot being deployed on a bridge by a quadcopter, which could take care of the localization. Secondly, it will be assumed that only straight path traversal will be required for the most part, with turning used only for path correction.

There are three requirements of movement that will be designed around:

1. The robot will need to be able to go in a roughly straight path, so that the robot can traverse quickly without continuously making sharp turns to correct its straight path
2. The robot will need to be able to detect edges so it won't fall off the bridge
3. The robot will need to be able to identify fairly accurately the locations that structural health measurements on the bridge are taken at to about 2-4 m resolution as was done in the experiment done with the previous version of the robot [4].

4.1.3.1 Requirement 1: Straight path traversal

In order to help the robot traverse roughly a straight path, the team implemented solely encoders for the scope of this project as its only downside of drift is minimal. The team also incorporated a gyroscope into the design and have the gyroscope's data accessible, but the actual incorporation of the data into path correction is a goal for the future. The Encoder solution pros and cons for straight path traversal are in in Table 3.

TABLE III

ENCODER PROS AND CONS

Name/Description	Pros	Cons
Encoders: **Measures rotations of wheels**	• Cheap • Simple data to process • Probably not much drift due to permanent magnets holding the wheels well the to the bridge • Can be used for accurate path correction by measuring how much one wheel should spin over the other	• Still potential for drift since there is no absolute reference

4.1.3.2 Requirement 2: Edge Detection

To accomplish edge detection, the team used IR sensors in the configuration shown in Figure 10. As shown in this figure, the IR sensor will be before the front-most edge of a wheel so it won't hit into inclines, but it will be past the floor-touching part of the wheel to ensure the IR sensor detects edges before the robot falls off. The pros and cons of IR sensors are listed in Table 4 below.

TABLE IV

IR SENSORS PROS AND CONS

Name	Pros	Cons
IR Sensors	• Have been implemented before • Simple data to process • Cheap	• Color of bridge could affect edge detection (may need to adjust thresholds for an edge "hit" from bridge to bridge)

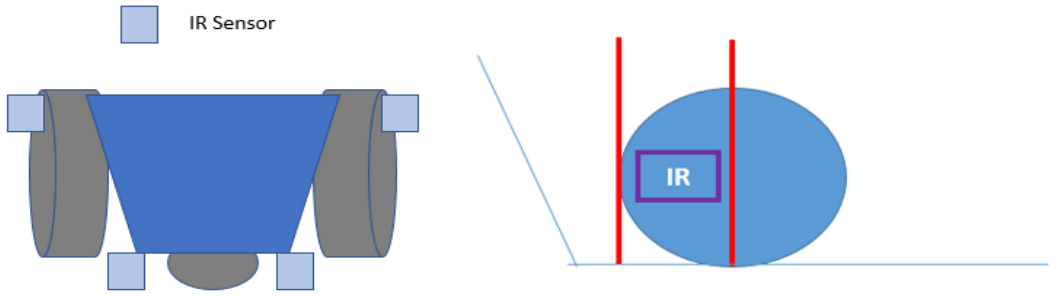


Figure 5. Diagram showing roughly the IR sensor placements on the robot, which is not drawn to scale (left). Diagram showing the placement of the IR sensor in a wheel (right)

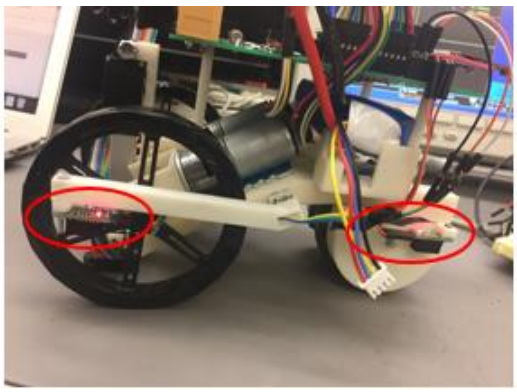


Figure 6. Picture showing IR sensors circled in red on one side of the robot.

4.1.3.3 Requirement 3: Motion Tracking

To accurately take structural health measurements, the robot design must be able to fairly and accurately assess the location where accelerometer measurements are taken. The team implemented motion tracking solely through encoders and this approach's pros and cons are listed in Table 5. The GPS data is accessible but hasn't been incorporated into the motion tracking algorithm. In the future, GPS data can be used to recalibrate the encoders every 50 m with 6% certainty.

TABLE V

ENCODER PROS AND CONS FOR MOTION TRACKING

Name	Pros	Cons
Solely Encoders	• Already on robot design for previous motion control and tracking requirements so can serve dual purpose • Probably not much drift due to permanent magnets holding the wheels well the to the bridge	• Still potential for drift since there is no absolute reference

4.1.3.4 GPS Selection

The team chose the Adafruit Ultimate GPS Breakout - 66 Channel MTK3339 with its features in Table 6 below. The GPS was not implemented in motion tracking for this prototype, but the team has proven data can be taken from the GPS for control later down the line.

TABLE VI

GPS FEATURES

Name	Price	Accuracy	Update Frequency	Sensitivity	Power	Other
Adafruit Ultimate GPS Breakout - 66 Channel MTK3339	$39.95	3 m	10 Hz	165 dBm	100 m	Comes with breakout board.

4.1.3.5 Gyroscope Selection

For gyroscope selection, the team chose the ST L3GD20H with its features in Table 7 below. The gyroscope was not implemented in motion tracking for this prototype, but the team has proven data can be taken from the gyroscope for control later down the line.

TABLE VII

GYROSCOPE FEATURES

Name	Price	Range/Resolution	Accuracy	Power Draw	Interface	Other
ST L3GD20H	$3.42	±245/±500/±2000°/s with 16 bits	Zero Bias: ±25°/s	15 mW	I2C/SPI	User enabled integrated low-p and high-pass filters. Temp sensor.

4.1.4 Magnets

The selected three-wheel design incorporates permanent neodymium magnets that wrap around the treads of the two front wheels, as shown in Figure 6. The K&J Magnetics B641 is used as a suitable magnet for this design. Each B621 magnet is 3/8” long by 1/8” wide by 1/16” thick, magnetized through the dimension of thickness, and has a pull force of 0.55 kg. Since the two front wheels are “magnetized”, and the back wheel has two lines of magnets, the combined pull force for the magnets is roughly 2.2 kg, which exceeds the weight specification of the robot.

4.1.5 Printed Circuit Board

The electronics of the robot are all routed through a single printed. The PCB provides header connections for all major components in addition to surface mounted power circuitry and filter capacitors. All major components besides the power circuitry is detachable for modularity. The board is designed in EagleCAD and printed with Advanced Circuits.

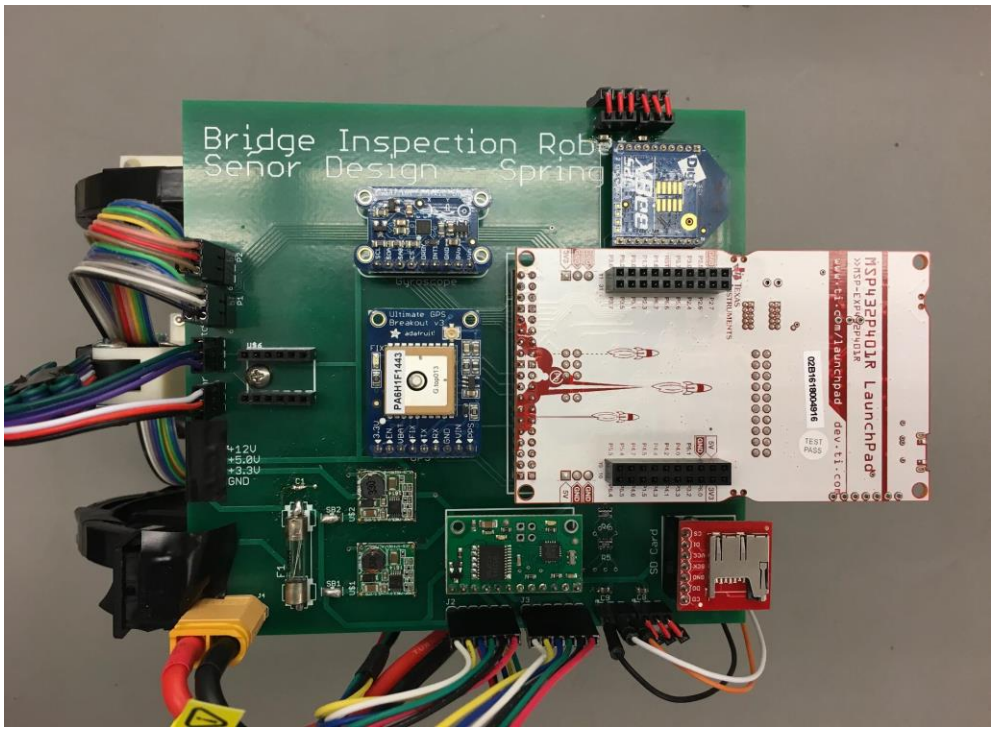


Figure 7. Finished printed circuit board

4.1.6 Structure

The supporting structure of the robot is entirely 3D printed. The design consists of several pieces designed in Autodesk Inventor. The pieces are attached together with M3 screws.

The main structure is a bent 4mm thick structure with mounting holes for all major pieces. In the front, two gear boxes are mounted to hold the drivetrain. At each gearbox, a motor sits along the length of the robot and the wheel at a 90-degree angle to it. Two bevel gears sit within the gearbox to transfer motion from the motor to the wheel. Between the two gearboxes is a mounting structure for the linear actuator. This structure holds the linear actuator perpendicular to the ground. Attached to end of the linear actuator is a housing for the accelerometer. Also 3D printed, this housing encases the evaluation board used while exposing the underside of the board and the accelerometer directly to the bridge for noise minimization. The topside of the housing allows for direct mounting to the linear actuator. In the back of the main structure are two supports for the back wheel. These are slightly smaller in radius that the back wheel to allow for the attachment of permanent magnets. Also included on these supports are mounting points for an IR sensor on either side of the back wheel. Coming out from the side of the main structure are two more supports for the front IR sensors. Two long beams jut out toward the front of the robot, each holding an IR sensor beside the front wheel. The beams are hollow, allowing for the sensors' wiring to be routed inside the beam, keeping it from getting caught in the wheel. At the center is a holding structure for the battery. A strap is still needed, however, to keep the battery from falling when the robot is upside-down. Lastly, there are three mounting holes on which stand three standoffs that hold the board 52mm above the main structure.

4.1.7 Microcontroller

The microcontroller of the Bridge Inspection robot is responsible for receiving input from sensors, sending data to the wireless transceiver, and controlling the robot's servos and motors. The team selected the TI MSP432P401R (MSP432) for the robot's microcontroller. The MSP432 contains a 48 MHz ARM 32-bit CPU, 256 KB of flash memory, 64 KB of SRAM, and a 14-bit analog-to-digital convertor (ADC) [9]. The MSP432 Launchpad, shown in Figure 11, is a development board with a built-in USB debugger and breakout pins, which allows for rapid prototyping. Early in development, the team tested the functionality of other components such as the accelerometer and IR sensors using the MSP432 Launchpad. The team decided to integrate the MSP432 Launchpad into the robot's printed circuit board as a mountable component, rather than incorporate a surface-mounted MSP432 chip into the board design. This was done to simplify integration and debugging.

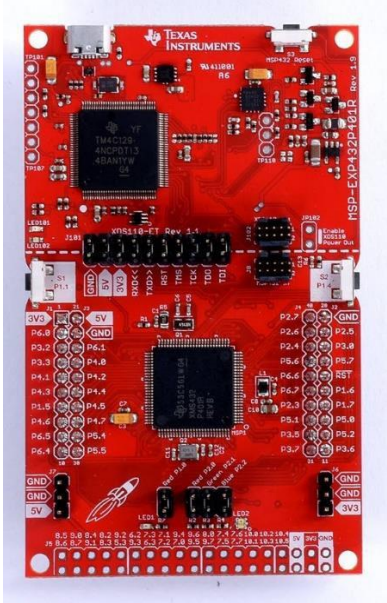


Figure 8. MSP432P401R Launchpad Development Kit*Bridge Structural Health Measurement*

4.1.1.1 Accelerometer

The accelerometer is the primary measurement tool of the Bridge Inspection Robot. At various locations along a bridge, the robot lowers its accelerometer to make contact with the support structures. The recorded structural vibrations show the overall frequency response and any changes in the bridge's natural frequency, which can then be factored into a decision for bridge repairs or other corrective actions [10]. Because the accelerometer is crucial to the project's goals, the team considered it a critical path item, and the accelerometer was supposed to be one of the first items the team would have purchased and tested.

The frequency response of bridges considered for inspection require a bandwidth of 0 – 50 Hz. Anything outside of this range is considered noise and not useful. The initial wheeled prototype version of the Bridge Inspection Robot used a Silicon Designs Low Power Single Axis Accelerometer Model 2012-002, which could read 0 – 300 Hz with a differential sensitivity of 2000 mV/g [11]. The updated Bridge Inspection Robot initially was expected to use a Silicon Designs Model 2460-002, an updated three-axis accelerometer differing from the Model 2012-002 only in size and axes of measurement [12]. The previous project advisor recommended a three-axis model due to significant lateral vibrations seen alongside measured vertical translations.

Figure 9. Silicon Designs Model 2460-002

After further discussion with Dr. Wang of the Civil Engineering department, the requirements for sensitivity were relaxed. The Model 2460-002 boasted 10 µg/√Hz for its noise specifications which, though highly noiseless, would have suffered from further noise from the electronics due to it being analog. This led to the design team switching to the Analog Devices ADXL355 Digital Accelerometer.

Figure 10. ADXL355-EVAL

The ADXL355 is still a triaxial accelerometer. Its noise specification was 25 µg/√Hz which, though more than double the specification of the 2460-002, did not suffer as much from surrounding electronics. Furthermore, filtering data from this accelerometer could be done in post-processing. This allowed for the Signal Conditioning Module of the original design to be unnecessary. The ADXL355 used came soldered onto an evaluation board with all the necessary components populated for ease of access to the pins. The board was mounted to the robot upside-down, however, so that the actual accelerometer would be directly touching the surface that is being measured. This is adjusted in post-processing by negating the z-axis.

As the accelerometer must be lowered onto the bridge, various methods of deployment were examined. The team settled on the Actuonix L12-I 30mm linear actuator which was a prepackaged solution that allowed for controllable vertical deployment of the accelerometer by pulse-width modulation or RC servo control by means of an in-built control board.

Figure 11. Actuonix L12-I with 30mm Stroke

Initially, it was expected that the built-in potentiometer could be used to see when the accelerometer made contact with the bridge. The idea was that the voltage proportional to the position of the linear actuator would stop changing when the accelerometer stopped moving. The issue that arose, however, was the 210:1 gearing ratio of the linear actuator meant it was strong enough to lift the robot off a metal surface even when magnetized. The solution to this was to set the value of the actuation to the level of the bottom of the wheels, guaranteeing contact.

4.1.1.2 Digital Filtering

The team decided that the robot would collect raw accelerometer data and digital filtering would be done after extracting the data from the robot. Matlab functions, firceqrip() and firgr(), were found to be useful in filtering the accelerometer data with a FIR (Finite Impulse Response) scheme with linear phase delay.

4.1.2 SD Card

The robot interfaces with a Sparkfun microSD breakout board, pictured in Figure 12, via SPI to save accelerometer data. The MSP432 is configured to read and write to the microSD card at 1 MHz. A Sandisk Class 4 16 GB microSD card provides storage for over 370 hours of data sampled from the accelerometer at 1000 Hz.

Figure 12. Sparkfun microSD Breakout Board

4.1.3 Wireless Communication

During operation, the robot communicates wirelessly with a PC through a dedicated module. The team desired a wireless communication module that would meet the following specifications: a maximum range of (at least) 800 m, a throughput greater than 10 Kbps, current consumption of less than 50 mA, and ability to interface via Serial Peripheral Interface Bus (SPI). The team identified the XBee S2C 802.15.4 module, pictured in Figure 13, as the best wireless transceiver for the Bridge Inspection Robot [13]. The XBee S2C 802.15.4 offers a maximum outdoor range of 1200 m, a throughput of up to 250 Kbps, current draw of around 30 mA, and the ability to interface with a microcontroller through UART or SPI. The team chose UART for interfacing with the XBee because of the simplicity in configuration compared to SPI. To complete the wireless link, a second module is connected to a PC through a USB breakout board also shown in Figure 13.

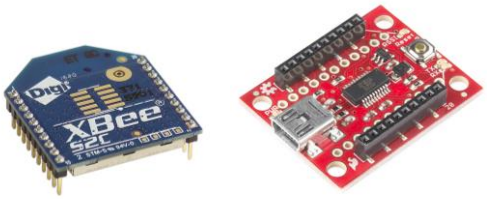


Figure 13. XBee S2C 802.15.4 Module(left) and Sparkfun XBee USB Explorer (right)

4.1.4 Batteries and Power

The robot is expected to run actively for at least one hour, traversing the bridge while making measurements, and an additional hour passively, staying in place while transmitting data. The battery chosen provides a suitable capacity to power the major components of the robot. These components include the motors, linear actuator, MCU, data acquisition, data transfer, and various supporting electronic circuitry. Of these, the highest minimum voltage needed is roughly 12V, for the motors, linear actuator, and accelerometer.

Each motor has a stall current of 1.1A and free run current of 100mA. Because of the light weight of the robot, the motors will run far below the stall current. The highest expected nominal draw, when traversing vertically, is 800mA each. The linear actuator will only be used for short movements of a small mass and thus will have low consumption. In conjunction with the accelerometer, expected current draw is only 35mA. The MCU and wireless module each require an input voltage of 3.3V. The accelerometer. The wireless module, MCU, GPS, and gyroscope are expected to collectively consume 300mW at 3.3V. With a power conversion efficiency of 80%, this would draw roughly 32mA from the battery. The remaining circuitry will have negligible power consumption. The final sum of power consumption with some additional margin of error is 1700 mA with maximum power consumption while active movement, and only 100 mA while passive. Therefore, a battery capacity of at least 1800mAh is required to power the robot for one hour actively and one hour passively.

TABLE VIII

BATTERY SPECIFICATIONS

Battery Type	Voltage Rating	Capacity	Recharge?	Weight	Vendor
Lipo 3S	11.1V	2200mAh	Yes	170g	Turnigy

A three-cell LiPo battery was chosen to be used to power the robot. This battery will provide 2200mAh at 11.1V, satisfying the design constraints but also offering rechargeability and high current output for instantaneous torque from motors, all within a package that is lightweight compared to its output.

4.1.5 External Server (PC)

A PC connected to an XBee 802.15.4 module through a USB dongle serves as the "master" for the Bridge Inspection Robot. A simple command-line interface, shown in Figure 14, allows a user to send commands over a serial connection at a baud rate of 115200; these commands direct the robot to move forward along the bridge and record accelerometer data. Additionally, a MATLAB program can be run on the PC to plot accelerometer data that is transmitted wirelessly from the robot.

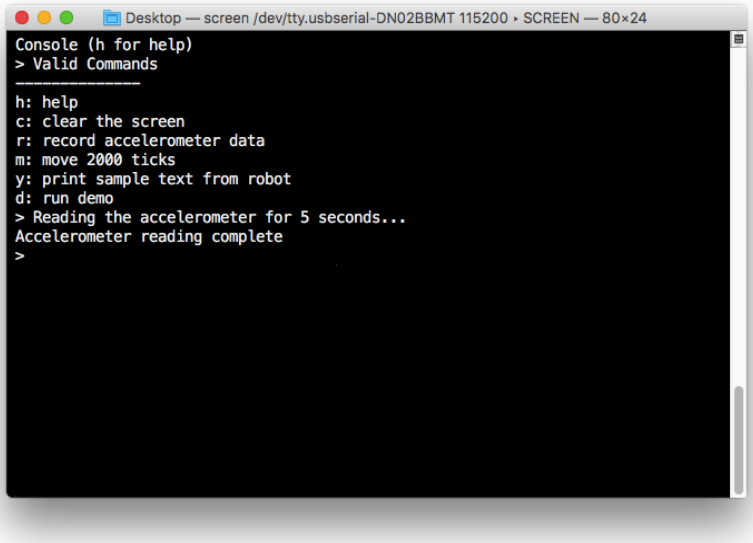


Figure 14. Example of Command-Line Interface In Use

4.1.6 Software

All the code that runs on the robot's MSP432 is written in C and compiled through TI's IDE, Code Composer Studio. Because many of the robot's components have to run simultaneously, the MSP432 is configured to run on TI-RTOS, a proprietary RTOS optimized for TI microcontrollers. TI-RTOS comes with built-in libraries for many microcontroller functions, such as GPIO, ADC, timers, etc., that simplify embedded development. The robot's functions are divided amongst several threads, one for each major component on the robot. Mutexes, semaphores, and barriers sync and coordinate actions between these threads.

Figure 19 shows the C and header files that are part of the robot's program. Many of the threads are based upon examples provided by TI through their Resource Explorer tool. Libraries for the motors and GPS are available from their respective companies for the Arduino platform; these were ported over to the MSP432 by the team.be run on the PC to plot accelerometer data that is transmitted wirelessly from the robot.

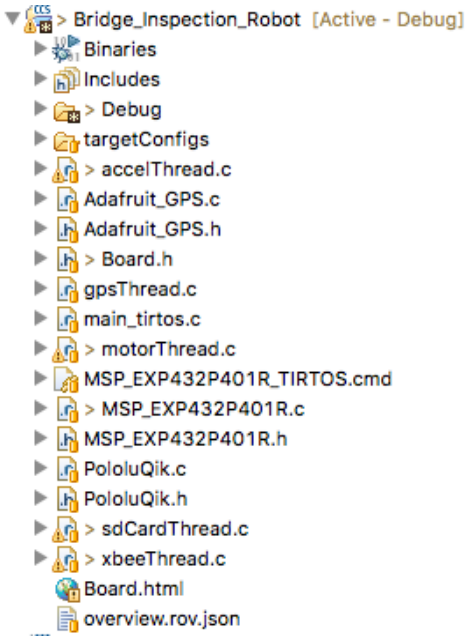


Figure 15. Bridge Inspection Robot Program Files

4.1.6.1 User Interface

xbeeThread.c creates a command-line interface for controlling the robot. Upon initialization, the robot waits to receive a character from the PC via XBee. A switch statement *inside xbeeThread.c* maps the received character to a command for the robot. (Psuedocode for the user interface is given in Figure 16). For example, to send a text sample from the robot to the PC (for the purpose of testing connection strength), the user types ‘y’ into the serial terminal. See Figure 16 for an example of the interface running.

```
48  while (1) {
49      UART_read(uartXbee, &cmd);
50      switch (cmd) {
51          case 'r':
52              UART_write(uartXbee, "Starting accel reading...\r\n");
53              recordAccel(1000, 5); // 1000 Hz, 5 seconds
54              UART_write(uartXbee, "Accel reading complete\r\n");
55              break;
56          case 'm':
57              UART_write(uartXbee, "Moving robot forward...\r\n");
58              moveMotors(2000); // 2000 encoder ticks
59              UART_write(uartXbee, "Movement complete\r\n");
60              break;
61          case 'd':
62              UART_write(uartXbee, "Running demo...\r\n");
63              runDemo();
64              break;
65          case 'y':
66              sendSampleOutput();
67              break;
68          default:
69              UART_write(uartXbee, helpPrompt);
70              break;
71
72      }
73  }
```

Figure 16. Switch Statement for User Interface Commands

4.1.6.2 Datalogging Program

The process of reading the accelerometer and saving the values to the SD card is handled by *accelThread.c* and *sdCardThread.c*.

The accelerometer thread configures the accelerometer's sampling rate (referred to as "output data rate" in ADXL355 documentation) to 1000 Hz. A FIFO (queue) on the accelerometer is configured so that an interrupt is triggered when 24 samples of data are available to be read over SPI; this enables the MSP432 to read the accelerometer in bursts, reducing overhead time. A sample is a 20-bit signed integer value for either the x-, y-, or z-axis; each sample is cast to a 32-bit signed integer as the data is read. These samples are saved to one of two statically-allocated buffers, each with a size of 1536 samples. When one buffer is filled, a semaphore signals the SD card thread to begin writing that buffer to an open file on the SD card.

The SD card thread writes 512 bytes of data to the SD card at a time. The accelerometer thread signals the SD card thread to perform one such write after every 96 samples are read. These writes are scheduled so that the accelerometer is not left idle for too long, which results in an overflow of the FIFO and the loss of samples. Furthermore, this configuration ensures that the SD card finishes reading its buffer before the accelerometer fills up its buffer, thus preventing buffer overwrites. Figure 17 shows the accelerometer thread's code for reading the accelerometer and storing the values in a buffer.

```
for (i=0; i<ACCEL_DATA_BUF_COUNT; i+=ACCEL_WATERMARK_SAMPLES) {
    while (!GPIO_read(6));  // Wait for 24 samples to be available
    for (j=i; j<i+ACCEL_WATERMARK_SAMPLES; j+=3) {
        GPIO_write(4, 0);
        transferOK = SPI_transfer(masterSpi, &masterTransaction);
        GPIO_write(4, 1);
        accelDataBuffer1[j] = ((accelRxBuffer[1] << 24)
                             | (accelRxBuffer[2] << 16)
                             | (accelRxBuffer[3] << 8)) >> 12;
        accelDataBuffer1[j+1] = ((accelRxBuffer[4] << 24)
                               | (accelRxBuffer[5] << 16)
                               | (accelRxBuffer[6] << 8)) >> 12;
        accelDataBuffer1[j+2] = ((accelRxBuffer[7] << 24)
                               | (accelRxBuffer[8] << 16)
                               | (accelRxBuffer[9] << 8)) >> 12;
    }
}
```

Figure 17. Pseudocode for Reading the ADXL355

The accelerometer thread also handles the linear actuator, which is controlled like an RC servo. Prior to an accelerometer reading, a PWM signal is sent to the linear actuator to deploy the accelerometer. When the reading is complete, another PWM signal is sent to retract the accelerometer.

4.1.6.3 Movement Program

Forward movement of the robot is handled by *xbeeThread.c* and *motorThread.c*, which contains a controls algorithm for keeping the wheels straight, as well as interrupts for the IR sensors and motor encoders. The motors are controlled via the *PololuQik_setSpeeds()* function, which takes integer values from -128 to 128 to control the speed and direction for the two motors. When the motor thread receives a semaphore signal from the XBee thread, the encoder counts for Motor 0 and Motor 1 are reset to 0, and the motors are initially set to a speed of 64 for Motor 0, and -64 for Motor 1, so that the robot begins moving forward. (To move the robot backwards, reverse the signs of these values).

Each motor has a quadrature encoder that provides two output signals. When a rising edge on either output signal is detected, the corresponding interrupt function is called to determine whether the motor is moving forwards or backwards, and increments or decrements the motor's encoder count accordingly. The motors stop moving when either encoder count passes the distance threshold.

As the robot moves forwards (or backwards), a timer running in xbeeThread.c causes the motor thread to update the motor speeds every 0.1 seconds. Inside the motor thread is a proportional control algorithm that reads copies of the encoder counts, finds the difference between the two counts, and increases the speed of the motor whose wheel is lagging behind. (Copying the encoder counts allows the original counts to keep updating while the speeds are changed.) At the same time, the XBee thread prints the encoder counts and new motor speeds to the serial terminal.

Each IR sensor has its own interrupt function that is triggered whenever the IR sensor detects an edge. These functions set a flag in the motor thread that causes it to stop the motors and encoders and then perform certain hard-coded movements, dependent upon which IR sensor set the flag, to correct the robot's direction.

Although the GPS works with the MSP432, and the absolute location of the robot can be retrieved from the GPS, GPS data was not incorporated into the distance measurement code. This is an area to address in future development.

Pseudocode for an IR sensor interrupt and the motor differential code is given in Figure 18.

```
19  void edgeDetected(uint_least8_t index) {
20      if (!irFlag) {
21          irFlag = index; // irFlag indicates where edge is
22      }
23  }
24  // ...
25      while (keepMoving) {
26          if (irFlag) {
27              correctForEdge(irFlag);
28          }
29          error = enc1Count - enc0Count;
30          if (error > 0) { // increase m0 speed
31              m0Speed = 64 + error/3;
32              m1speed = 64;
33              if (m0Speed > 90) {
34                  m0Speed = 90;
35              }
36          } else {        // increase m1 speed
37              m0Speed = 64;
38              m1speed = 64 - error/3;
39              if (m1speed > 90) {
40              m1speed = 90;
41              }
42
43          }
44          PololuQik_setSpeeds(uartMotor, m0Speed, -m1speed);
45          sem_wait(&semMoveMotors);   // wait 0.1 seconds
46      }
```

Figure 18. Pseudocode for IR Sensor Interrupt and Motor Differential Code

4.2 Codes and Standards

One of the most significant standards for this project is Serial Peripheral Interface Bus (SPI). This is a synchronous bus interface protocol used to send data between device components [14]. It will be needed to interface the MSP432 with the ADXL355 accelerometer, the SD card, and the gyroscope. SPI is a straightforward protocol, requiring only four wires to implement. Furthermore, the intricacies of implementing SPI is abstracted away by open- source MSP432 code. However, understanding the constraints of SPI communication is critical for addressing the robot's design needs. For example, SPI is a single-master protocol, meaning that only one device on the SPI network can send commands to other devices. For the Bridge Inspection Robot, the MSP432 serves as the SPI master.

IEEE 802.15.4 is a protocol defined for low-rate wireless personal area networks. It is a point-to-multipoint protocol that, unlike mesh protocols, ensures that all other nodes in the network receive commands from a sending node at the same time, provided that they are all within range of the sending node. This is critical for future development, because a network of multiple Inspection Robots must be synchronized with each other to function properly. The XBee 2SC 802.15.4 operates on the 2.4 GHz range.

The MSP432 is programmed in C, a low-level programming language that is most popular for small-scale embedded systems programming.

4.3 Constraints, Alternatives, and Tradeoffs

The current design of the robot can traverse vertically and upside-down only on steel bridges since it uses magnets for these kinds of traversals. It must be small enough to stay on and traverse the support structures. Furthermore, the total weight of the robot is limited by the holding weight of the magnets, the servos, and the quadcopter used for delivery.

For data measurement, the accelerometer choice is dictated by the frequency of the vibrations expected, which are all below 30 Hz. The robot must also be able to take multiple measurements within a single run, meaning the battery and magnet designs must support the expected operation time of three to four hours before recharge. Finally, as the robot will be sending measurement data wirelessly, the choices concerning wireless communications are dictated by an expected maximum operation distance of 800 m away from an external computer.

An alternative to a mobile network is a static network of wireless sensors along the bridge. However, the accelerometers that are needed for accurate measurements typically cost several hundred dollars, making it unaffordable to densely equip bridges with a large number of sensors. Using a small number of sensors on the other hand results in poor spatial resolution that does not provide high enough accuracy for damage detection. A mobile network allows the robots to deploy in a tight configuration that allows for high resolution during data collecting, and then dynamically reconfigure to another part of the bridge to repeat this process.

Balancing performance and power consumption is the most significant tradeoff for the Bridge Inspection Robot. Several design decisions were made that compromise the robot's speed in favor of extending the robot's battery life. For example, a microprocessor would provide faster computing performance than a microcontroller, and it would allow the robot to extend its functionality with more

computationally- intensive components, such as a camera. However, the higher power requirements of a microprocessor and a camera would significantly reduce the battery life.

5 Schedule, Tasks, and Milestones

The Gantt chart in Appendix A shows the tasks that the team completed. For each specific task this chart outlines major milestones, start dates, end dates, and durations. This chart also visually shows the timeline for these tasks. Appendix B shows the team's Pert chart, which displays the major components of the project along with their associated start date, end date, duration, critical path in blue, and percentage completed. Sanmesh and Erikzzon wer responsible for the Bridge Structural Health Measurements tasks, Kristen and Justin were responsible for the Sensing Environment tasks, Sanmesh, Erikzzon and Sean were responsible for the Mechanical Design tasks, Justin and Kristen were responsible for the Wireless Communication tasks, and Sean, Erikzzon, and Sanmesh were responsible for the Movement tasks.

6 Final Project Demonstration

For the final demonstration, the external PC is connected to an XBee acting as the host. Using MATLAB, the data is read through a serial port connection. With the baud rate set to 115200, the transmission time is roughly 10 seconds for a standard measurement: 5 seconds of data sampled at 1000 Hz. The data is received as a comma delimited set of X,Y,Z data in units of 3.8ug. The data is converted in to g’s and plotted both in the time domain and frequency domain.

7 Marketing and Cost Analysis

7.1 Marketing Analysis

There are a few wireless bridge structural health monitoring systems that are comparative to the Bridge Inspection Robot.

The SensSpot is a sensor that can be placed on bridges using its self-adhesive property [16]. It has a minimum expected life of 20 years, and for an average-sized highway bridge, would need about 500 sensors each $20 for a total of $10,000 [17]. The Bridge Inspection Robot's sensing nodes however would not need a mass deployment like the SensSpot because the mobile nature of the robot would allow a small number of sensing nodes to take measurements of the whole bridge over time. The team's robot also avoids the time and labor required in installing the SensSpot sensors. Lastly, the Bridge Robot would never run out of power during operation because it could always come back for charging during inactive times.

The robot described in the “Wireless Mobile Sensor Network for the System Identification of a Space Frame Bridge” by Dapeng Zhu et Al. is very similar to the Bridge Inspection Robot design, with the same mobile health measuring method at its base [10]. However, the key difference in the Bridge Robot’s design is that its motion mechanism allows it to move in multiple directions along the surface of a bridge while the robot described in [10] only allows for straight-line movement.

7.2 Cost Analysis

The total cost of the Bridge Inspection Robot component is estimated to be roughly $450. Table 9 shows a breakdown of the material costs of the prototype. It will have several sensors and actuators which will need to be purchased. The supporting structure for the robot can be designed in CAD software and 3D printed at a very low cost, and provide a very low weight structure that could be rapidly prototyped. A handful of miscellaneous circuit components will be needed to support the main chips, including capacitors, resistors, and power convertors. These, in addition to assembly pieces such as screw, will be estimated in price. The completed circuit will be printed professionally by a board house. This cost is estimated at $33 for a two layer board.

Table IX

Total Component Costs for Prototype

Item	Unit Price	Qty.	Total Price
Battery	$9.99	1	$9.99
GPS	$39.95	1	$39.95
Gyroscope	$12.50	1	$12.50
Microcontroller	$12.99	1	$12.99
Wireless Module Dev Kit	$59.95	1	$59.95
SD Card and Breakout	$10.94	1	$10.94
Accelerometer	$43.75	1	$43.75
Linear Actuator	$90.00	1	$90.00
IR Sensor	$6.95	4	$27.80
Pololu Metal Gearmotor Pair	$7.45	1	$7.45
Magnet	$0.21	150	$31.50
PCB Printing	$33.00	1	$33.00
Omni-Wheel	$9.95	1	$9.95
DC Converters	$3.45	2	$6.90
Bore Shaft Mount Bevel Gears	$5.99	4	$23.96
Screws/Wires/Misc	$15.00	1	$15.00
Caps/Res/Power Conv	$15.00	1	$15.00
		Total Cost:	**$450.63**

The labor costs are assumed to be at a rate of $20 per hour. At this rate, we find a total labor cost of $13,600. The breakdown of the labor costs is shown in Table 10.

Table X

Total Labor Costs for Development

Project Component	Labor Hours	Labor Costs
Structural Health Measurement	150	$3,000.00
Sensing Environment	150	$3,000.00
Mechanical Design	100	$2,000.00
Wireless Communication	80	$1,600.00
Movement	125	$2,500.00
Documentation/Reports	50	$1,000.00
Full Assembly Testing	25	$500.00
Total Labor Cost	**680**	**$13,600.00**

Using the fringe benefit as 30% of total labor and overhead as 120% of material and labor, the total development cost would be $39,886 The breakdown of these costs are shown in Table 11.

TABLE XI

TOTAL DEVELOPMENT COSTS

Description	Cost
Parts	$450
Labor	$13,600.00
Fringe Benefits, % of Labor	$4,080.00
Subtotal	$18,130
Overhead, % of Parts, Labor and Fringe	$21,756.0
Total	$39,886

8 Conclusion

The project has all of the necessary hardware components assembled for bridge traversal and accelerometer deployment. A simple command line interface has been created to wirelessly send movement and data collection commands to the robot. During movement, the robot is capable of edge detection and path correction. Through the team's verification tests of the robot, the team can conclude that the robot could be used for traversal and data collection on an actual bridge in sunny day conditions.

The robot's magnetic wheels prevent slippage on inclines and allow the robot to traverse steel surfaces in vertical and inverted configurations. Accelerometer data can be accurately collected and transmitted over a distance of at least 800m, which is the length of an average bridge. Once received on at the main computer, this data can be piped to a MATLAB program for automatic plotting. One thing to consider is that the transmission gives out if there is not a clear line of sight; this could be a problem if the robot is under the bridge and trying to transmit data. The batteries on the robot were chosen to ensure that the robot can traverse the whole bridge and take all the proper measurements without risking the battery dying. The PCB that was created for power, data transmission, and motor control implements a modular design so that major components can be detached to ease development or replaced with new parts.

The team also incorporated a GPS and a gyroscope into the robot's design, although the robot does not currently utilize these parts. However, the robot can read data from these sensors, and this data could be used to aid in minimizing drift for straight path traversal robot.

The physical structure of the robot can also be improved through improved designs that are more durable. Extended traversal on metal surfaces would occasionally result in some small mechanical parts of the robot becoming loose or falling off. Another consideration is that the robot did not meet the 1 kg weight requirement that was originally set; the robot had a final weigh in of 1.084 kg. A future goal for this

project is to have the robot modified such that it can be deployed and retrieved by drone, which is more feasible if the robot has a smaller mass.

9 References

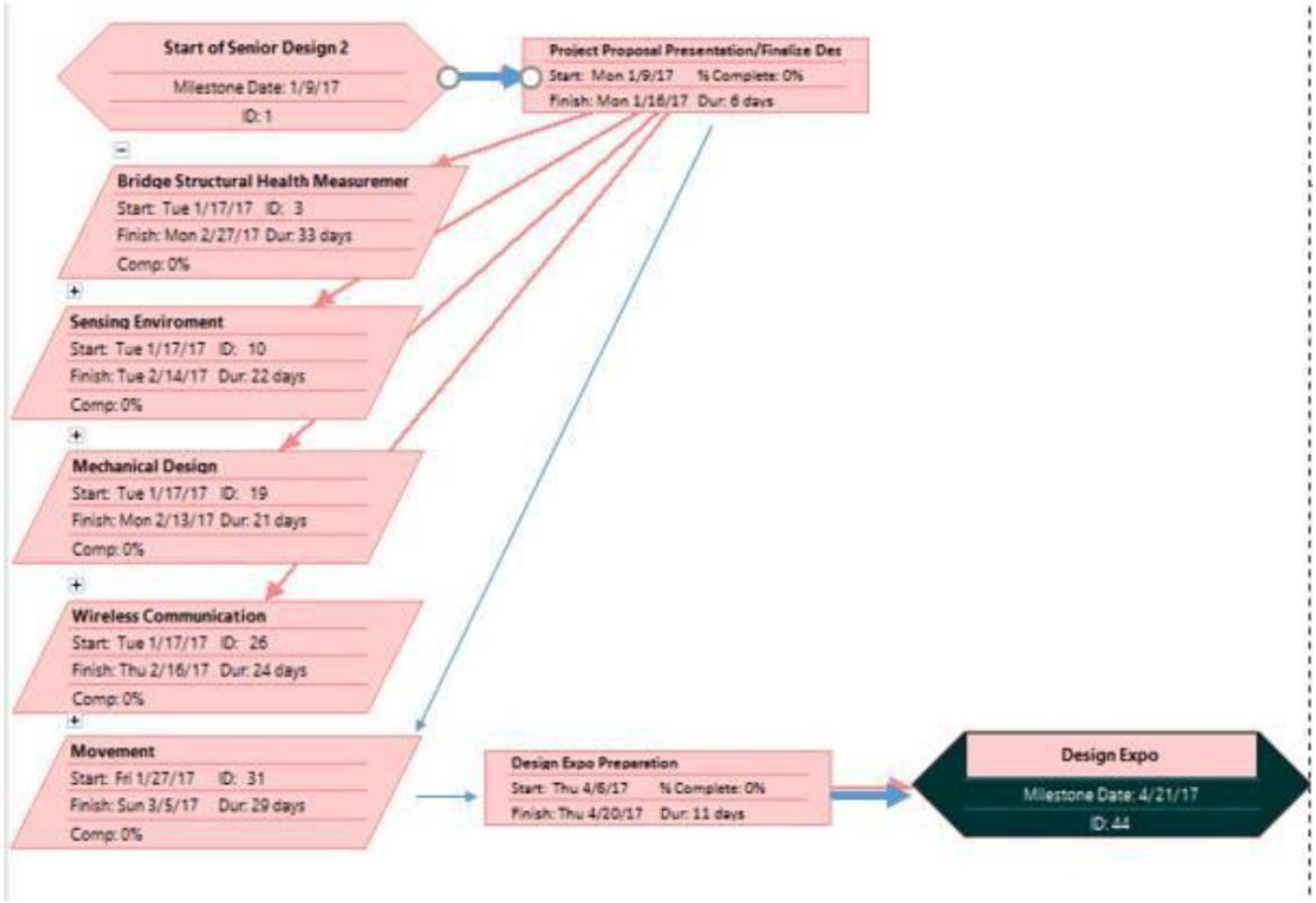
1. [1] M. Moore, B. Phares, B. Graybeal, D. Rolander, and G. Washer, "Reliability of Visual Inspection for Highway Bridges," Federal Highway Administration, McLean, VA Report No. FHWA-RD-01-020, 2001.
2. [2] J. M. Ko and Y. Q. Ni, "Technology developments in structural health monitoring of large-scale bridges," Engineering Structures, vol. 27, pp. 1715-1725, 2005.
3. [3] E. G. Straser and A. S. Kiremidjian, "A Modular, Wireless Damage Monitoring System for Structures," John A. Blume Earthquake Eng. Ctr., Stanford University, Stanford, CA Report No. 128, 1998. [4] Zhu, D., Guo, J., Cho, C., Wang, Y., and Lee, K.-M. (2012). "Wireless mobile sensor network for the system identification of a space frame bridge," Mechatronics, IEEE/ASME Transactions on, 17(3): 499-507.
4. [5] F. Tache, W. Fischer, G. Caprari, R. Siegwart, R. Moser, and F. Mondada, "Magnebike: a magnetic wheeled robot with high mobility for inspecting complex-shaped structures," Journal of Field Robotics, vol. 26, pp. 453-476, May 2009.
5. [6] M. P. Murphy and M. Sitti, "Waalbot: an agile small-scale wall-climbing robot utilizing dry elastomer adhesives," Mechatronics, IEEE/ASME Transactions on, vol. 12, pp. 330-338, 2007.
6. [7] W. R. Provancher, S. I. Jensen-Segal, and M. A. Fehlberg, "ROCR: an energyefficient dynamic wall-climbing robot," Ieee-Asme Transactions on Mechatronics, vol. 16, pp. 897-906, Oct 2011.
7. [8] M. Todd, D. Mascarenas, E. Flynn, T. Rosing, B. Lee, D. Musiani, et al., "A different approach to sensor networking for SHM: remote powering and interrogation with unmanned aerial vehicles," in Proceedings of the 6th International Workshop on Structural Health Monitoring, Stanford, CA, 2007.

8. [9] Texas Instruments, “MSP432P401R, MSP432P401M Mixed-Signal Microcontrollers”, Datasheet, March 2015. [Revised May 2016]. [Online]. Available: http://www.ti.com/lit/ds/symlink/msp432p401r.pdf. Accessed Nov. 20, 2016.
9. [10] D. Zhu, J. Guo, C. Cho, Y. Wang and K. M. Lee, "Wireless Mobile Sensor Network for the System Identification of a Space Frame Bridge," in IEEE/ASME Transactions on Mechatronics, vol. 17, no. 3, pp. 499-507, June 2012. DOI: 10.1109/TMECH.2012.2187915
10. [11] Silicon Designs, “Low Power Single Axis Accelerometer Module, Model 2012”, Datasheet, October 2013. [Online]. Available: http://www.krone.co.jp/pdf/2012.pdf. Accessed Nov. 21, 2016.
11. [12] Silicon Designs, “Low Cost & High Performance 1-Axis DC Accelerometer Modules”, Datasheet, 29 April 2015. [Online]. Available: https://drive.google.com/open?id=0B4eVto02URUhQnl4d0oyenlYeW8. Accessed Nov. 21, 2016.
12. [13] "XBee DigiMesh 2.4", Digi International, Minnetonka, MN, 2016. [Online]. Available: https://www.digi.com/products/xbee-rf-solutions/modules/xbee-digimesh-2-4. Accessed Nov. 20, 2016.
13. [14] Byte Paradigm, "Introduction to I²C and SPI protocols – Byte Paradigm – Speed up embedded system verification", Byteparadigm.com, 2016. [Online]. Available: http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/. Accessed: Nov. 21, 2016.
14. [15] "Wireless Mesh Networking: Zigbee vs. Digimesh" Digi International, Minnetonka, MN, 2015. [Online] Available: https://www.digi.com/pdf/wp_zigbeevsdigimesh.pdf. [Accessed Nov. 20, 2016].

15. [16] Resensys, "SenSpot," Resensys, 2011. [Online]. Available: http://www.resensys.com/product1.html. Accessed: Nov. 20, 2016.

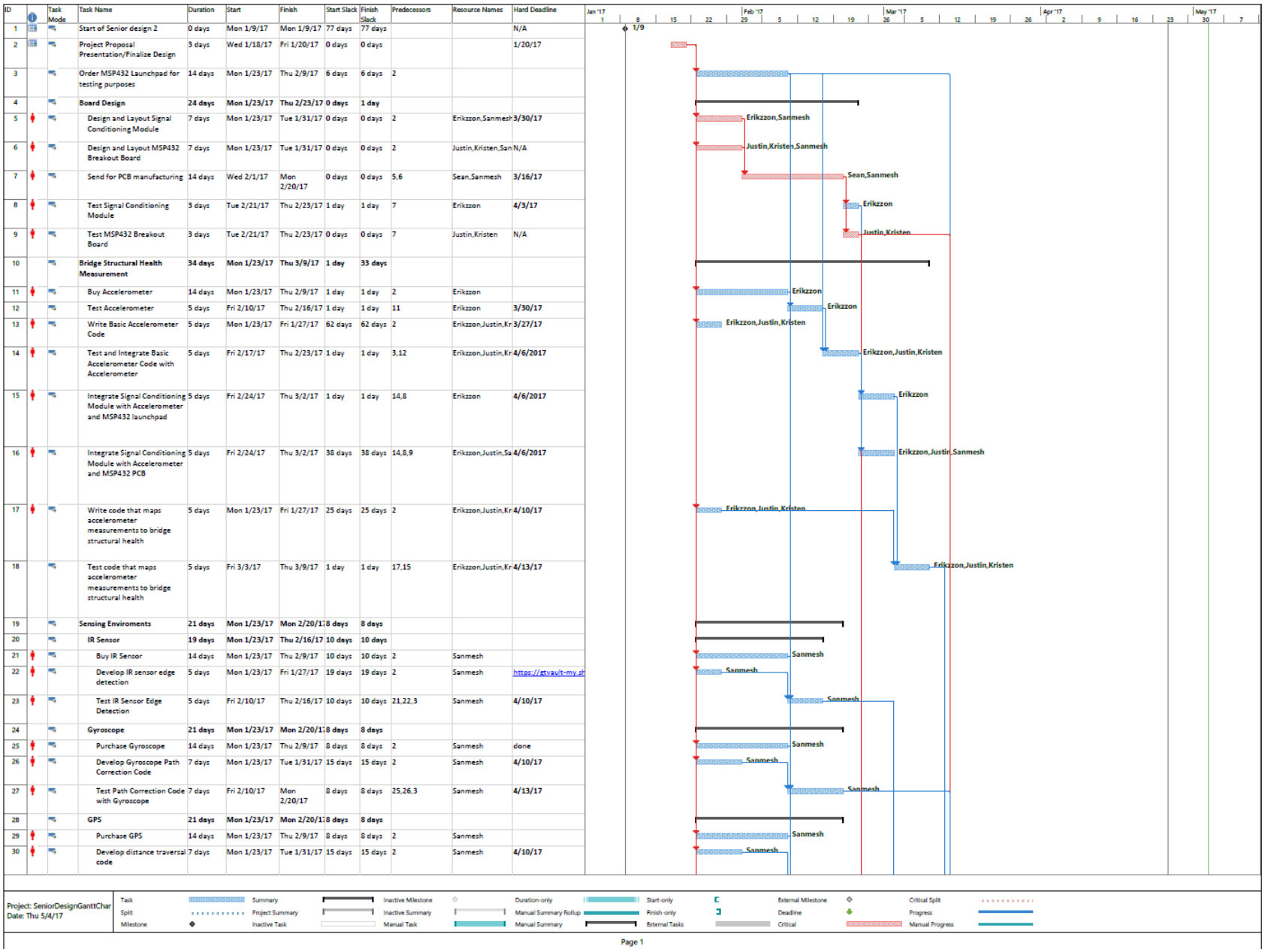
16. [17] D. Quick, "Wireless sensor to monitor structural integrity of bridges," New Atlas, 2011. [Online]. Available: http://newatlas.com/wireless-bridge-sensor/19380/. Accessed: Nov. 20, 2016.

Appendix A – PERT Chart

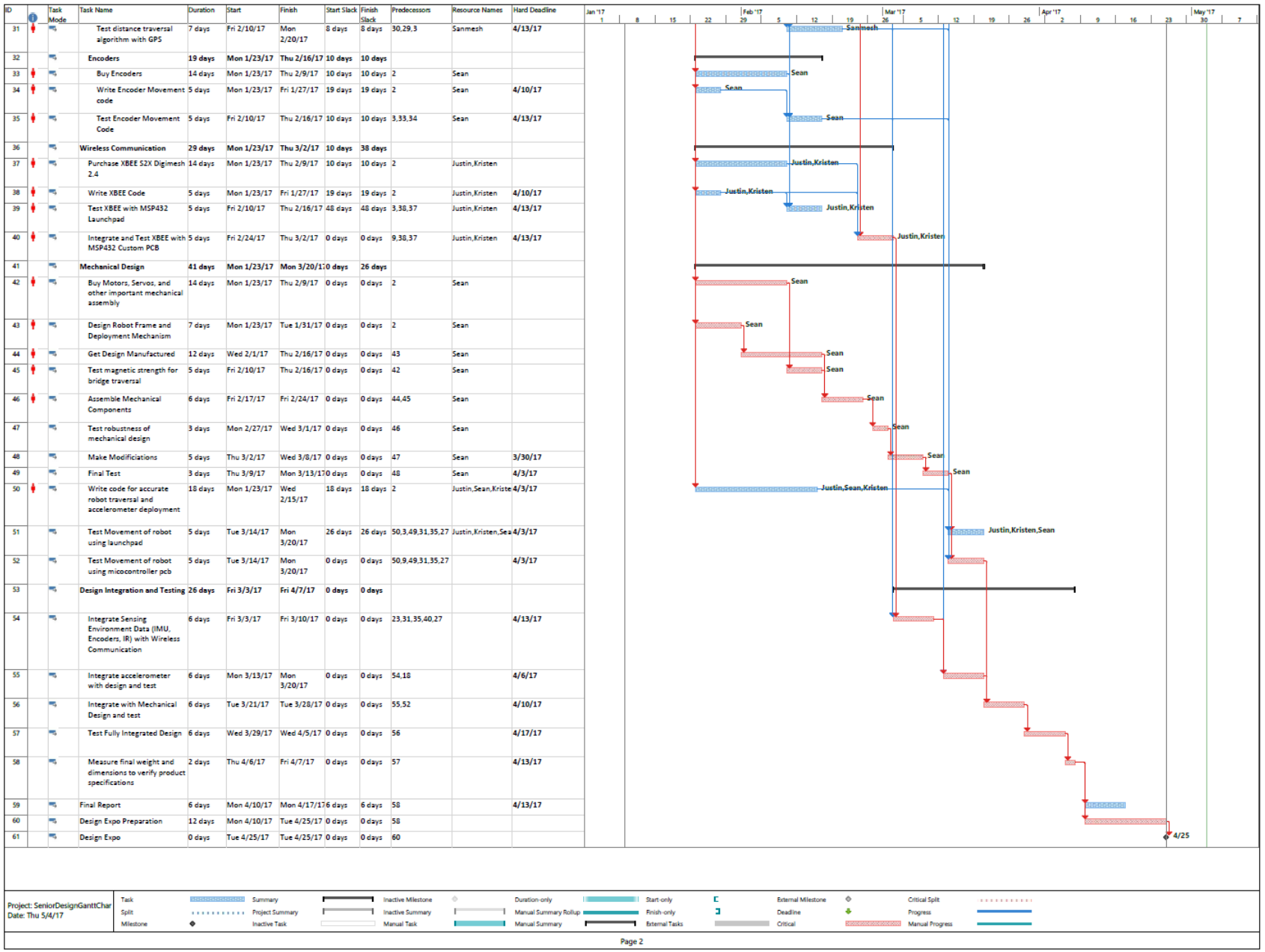


Appendix B – Gantt Chart

See next pages for project Gantt Chart.



ID	Task Mode	Task Name	Duration	Start	Finish	Start Slack	Finish Slack	Predecessors	Resource Names	Hard Deadline
1		Start of Senior design 2	0 days	Mon 1/9/17	Mon 1/9/17	77 days	77 days			N/A
2		Project Proposal Presentation/Finalize Design	3 days	Wed 1/18/17	Fri 1/20/17	0 days	0 days			1/20/17
3		Order MSP432 Launchpad for testing purposes	14 days	Mon 1/23/17	Thu 2/9/17	6 days	6 days	2		
4		**Board Design**	**24 days**	**Mon 1/23/17**	**Thu 2/23/17**	**0 days**	**1 day**			
5		Design and Layout Signal Conditioning Module	7 days	Mon 1/23/17	Tue 1/31/17	0 days	0 days	2	Erikzzon,Sanmesh	3/30/17
6		Design and Layout MSP432 Breakout Board	7 days	Mon 1/23/17	Tue 1/31/17	0 days	0 days	2	Justin,Kristen,San	N/A
7		Send for PCB manufacturing	14 days	Wed 2/1/17	Mon 2/20/17	0 days	0 days	5,6	Sean,Sanmesh	3/16/17
8		Test Signal Conditioning Module	3 days	Tue 2/21/17	Thu 2/23/17	1 day	1 day	7	Erikzzon	4/3/17
9		Test MSP432 Breakout Board	3 days	Tue 2/21/17	Thu 2/23/17	0 days	0 days	7	Justin,Kristen	N/A
10		**Bridge Structural Health Measurement**	**34 days**	**Mon 1/23/17**	**Thu 3/9/17**	**1 day**	**33 days**			
11		Buy Accelerometer	14 days	Mon 1/23/17	Thu 2/9/17	1 day	1 day	2	Erikzzon	
12		Test Accelerometer	5 days	Fri 2/10/17	Thu 2/16/17	1 day	1 day	11	Erikzzon	3/30/17
13		Write Basic Accelerometer Code	5 days	Mon 1/23/17	Fri 1/27/17	62 days	62 days	2	Erikzzon,Justin,Kr	3/27/17
14		Test and Integrate Basic Accelerometer Code with Accelerometer	5 days	Fri 2/17/17	Thu 2/23/17	1 day	1 day	3,12	Erikzzon,Justin,Kr	4/6/2017
15		Integrate Signal Conditioning Module with Accelerometer and MSP432 launchpad	5 days	Fri 2/24/17	Thu 3/2/17	1 day	1 day	14,8	Erikzzon	4/6/2017
16		Integrate Signal Conditioning Module with Accelerometer and MSP432 PCB	5 days	Fri 2/24/17	Thu 3/2/17	38 days	38 days	14,8,9	Erikzzon,Justin,Sa	4/6/2017
17		Write code that maps accelerometer measurements to bridge structural health	5 days	Mon 1/23/17	Fri 1/27/17	25 days	25 days	2	Erikzzon,Justin,Kr	4/10/17
18		Test code that maps accelerometer measurements to bridge structural health	5 days	Fri 3/3/17	Thu 3/9/17	1 day	1 day	17,15	Erikzzon,Justin,Kr	4/13/17
19		**Sensing Enviroments**	**21 days**	**Mon 1/23/17**	**Mon 2/20/17**	**8 days**	**8 days**			
20		**IR Sensor**	**19 days**	**Mon 1/23/17**	**Thu 2/16/17**	**10 days**	**10 days**			
21		Buy IR Sensor	14 days	Mon 1/23/17	Thu 2/9/17	10 days	10 days	2	Sanmesh	
22		Develop IR sensor edge detection	5 days	Mon 1/23/17	Fri 1/27/17	19 days	19 days	2	Sanmesh	https://gtvault-my.sh
23		Test IR Sensor Edge Detection	5 days	Fri 2/10/17	Thu 2/16/17	10 days	10 days	21,22,3	Sanmesh	4/10/17
24		**Gyroscope**	**21 days**	**Mon 1/23/17**	**Mon 2/20/17**	**8 days**	**8 days**			
25		Purchase Gyroscope	14 days	Mon 1/23/17	Thu 2/9/17	8 days	8 days	2	Sanmesh	done
26		Develop Gyroscope Path Correction Code	7 days	Mon 1/23/17	Tue 1/31/17	15 days	15 days	2	Sanmesh	4/10/17
27		Test Path Correction Code with Gyroscope	7 days	Fri 2/10/17	Mon 2/20/17	8 days	8 days	25,26,3	Sanmesh	4/13/17
28		**GPS**	**21 days**	**Mon 1/23/17**	**Mon 2/20/17**	**8 days**	**8 days**			
29		Purchase GPS	14 days	Mon 1/23/17	Thu 2/9/17	8 days	8 days	2	Sanmesh	
30		Develop distance traversal code	7 days	Mon 1/23/17	Tue 1/31/17	15 days	15 days	2	Sanmesh	4/10/17



ID	Task Mode	Task Name	Duration	Start	Finish	Start Slack	Finish Slack	Predecessors	Resource Names	Hard Deadline
31		Test distance traversal algorithm with GPS	7 days	Fri 2/10/17	Mon 2/20/17	8 days	8 days	30,29,3	Sanmesh	4/13/17
32		**Encoders**	**19 days**	**Mon 1/23/17**	**Thu 2/16/17**	**10 days**	**10 days**			
33		Buy Encoders	14 days	Mon 1/23/17	Thu 2/9/17	10 days	10 days	2	Sean	
34		Write Encoder Movement code	5 days	Mon 1/23/17	Fri 1/27/17	19 days	19 days	2	Sean	4/10/17
35		Test Encoder Movement Code	5 days	Fri 2/10/17	Thu 2/16/17	10 days	10 days	3,33,34	Sean	4/13/17
36		**Wireless Communication**	**29 days**	**Mon 1/23/17**	**Thu 3/2/17**	**10 days**	**38 days**			
37		Purchase XBEE S2X Digimesh 2.4	14 days	Mon 1/23/17	Thu 2/9/17	10 days	10 days	2	Justin,Kristen	
38		Write XBEE Code	5 days	Mon 1/23/17	Fri 1/27/17	19 days	19 days	2	Justin,Kristen	4/10/17
39		Test XBEE with MSP432 Launchpad	5 days	Fri 2/10/17	Thu 2/16/17	48 days	48 days	3,38,37	Justin,Kristen	4/13/17
40		Integrate and Test XBEE with MSP432 Custom PCB	5 days	Fri 2/24/17	Thu 3/2/17	0 days	0 days	9,38,37	Justin,Kristen	4/13/17
41		**Mechanical Design**	**41 days**	**Mon 1/23/17**	**Mon 3/20/17**	**0 days**	**26 days**			
42		Buy Motors, Servos, and other important mechanical assembly	14 days	Mon 1/23/17	Thu 2/9/17	0 days	0 days	2	Sean	
43		Design Robot Frame and Deployment Mechanism	7 days	Mon 1/23/17	Tue 1/31/17	0 days	0 days	2	Sean	
44		Get Design Manufactured	12 days	Wed 2/1/17	Thu 2/16/17	0 days	0 days	43	Sean	
45		Test magnetic strength for bridge traversal	5 days	Fri 2/10/17	Thu 2/16/17	0 days	0 days	42	Sean	
46		Assemble Mechanical Components	6 days	Fri 2/17/17	Fri 2/24/17	0 days	0 days	44,45	Sean	
47		Test robustness of mechanical design	3 days	Mon 2/27/17	Wed 3/1/17	0 days	0 days	46	Sean	
48		Make Modificiations	5 days	Thu 3/2/17	Wed 3/8/17	0 days	0 days	47	Sean	3/30/17
49		Final Test	3 days	Thu 3/9/17	Mon 3/13/17	0 days	0 days	48	Sean	4/3/17
50		Write code for accurate robot traversal and accelerometer deployment	18 days	Mon 1/23/17	Wed 2/15/17	18 days	18 days	2	Justin,Sean,Kriste	4/3/17
51		Test Movement of robot using launchpad	5 days	Tue 3/14/17	Mon 3/20/17	26 days	26 days	50,3,49,31,35,27	Justin,Kristen,Sea	4/3/17
52		Test Movement of robot using micocontroller pcb	5 days	Tue 3/14/17	Mon 3/20/17	0 days	0 days	50,9,49,31,35,27		4/3/17
53		**Design Integration and Testing**	**26 days**	**Fri 3/3/17**	**Fri 4/7/17**	**0 days**	**0 days**			
54		Integrate Sensing Environment Data (IMU, Encoders, IR) with Wireless Communication	6 days	Fri 3/3/17	Fri 3/10/17	0 days	0 days	23,31,35,40,27		4/13/17
55		Integrate accelerometer with design and test	6 days	Mon 3/13/17	Mon 3/20/17	0 days	0 days	54,18		4/6/17
56		Integrate with Mechanical Design and test	6 days	Tue 3/21/17	Tue 3/28/17	0 days	0 days	55,52		4/10/17
57		Test Fully Integrated Design	6 days	Wed 3/29/17	Wed 4/5/17	0 days	0 days	56		4/17/17
58		Measure final weight and dimensions to verify product specifications	2 days	Thu 4/6/17	Fri 4/7/17	0 days	0 days	57		4/13/17
59		Final Report	6 days	Mon 4/10/17	Mon 4/17/17	6 days	6 days	58		4/13/17
60		Design Expo Preparation	12 days	Mon 4/10/17	Tue 4/25/17	0 days	0 days	58		
61		Design Expo	0 days	Tue 4/25/17	Tue 4/25/17	0 days	0 days	60		