Design of a Smart Miniature Vehicle

Juliet E Kaiser, Purdue University Northwest

Kaiser graduated from Purdue University Northwest in 2018 with a degree in Electrical Engineering and Technology and is continuing her education to obtain a master's degree.

Prof. Omer Farook, Purdue University Northwest

Omer Farook is a member of the faculty of Electrical and Computer Engineering Technology at Purdue University, Nothwest. Farook received the diploma of licentiate in mechanical engineering and B.S.M.E. in 1970 and 1972, respectively. He further received B.S.E.E. and M.S.E.E. in 1978 and 1983, respectively, from Illinois Institute of Technology. Farook's current interests are in the areas of embedded system design, hardware-software interfacing, digital communication, networking, image processing, and biometrics, C++, Python, PHP and Java languages. He has a keen interest in pedagogy and instruction delivery methods related to distance learning. He has a deep commitment to social justice and in achieving economic and educational equity.

TABLE OF CONTENTS

ABSTRACT

The paper expounds the Senior Design II project which served the purpose of a capstone course for Electrical Engineering Technology program. The design is based on the utilization of ultrasonic and infrared sensors, microcontrollers, servo motors, a gyro sensor, Bluetooth, and C++ programming to create a small, remote controlled vehicle that moves at a user's instruction while maintaining its own well-being so as to prevent self-damage.

Elaborated in the paper is the design functionality of the project's robot for detecting upcoming events in terms of encountering objects, platform openings, or extreme tilting in its path that may cause harm if current trajectory is further continued. When such obstacles are detected, the vehicle's programming instructs it to cease movement, back-up as necessary, and override and disregard the user's instruction that would point it toward harm's way. Where a directed path is deemed hospitable, the vehicle follows the preprogramed instruction unconditionally.

The vehicle utilizes a Parallax Boe-Bot kit[1] chassis while the information processing and movement program is run through an Arduino Mega board; these separate components are joined via a fiberglass platform which also holds its sensor components. The device's remote controller utilizes an app to connect with the vehicles onboard Bluetooth module where forward movement and left and right turning commands are made accessible to the user as well as the option to stop the vehicle's movement if desired. The vehicle is powered by common batteries connected to the Arduino board via barrel jacks. Found in this paper is the vehicle's design framework, programming code, and experimentation data. While this project's vehicle hosts forward directional control, no reverse movement controls are made accessible to the remote-user so as to limit the required size of the vehicle and the required input sensor quantity to an experimental, while non-redundant, level for the purposes of this project's defined parameters. The paper goes into greater detail regarding these restraints and the project's value as a scaled-concept model to like-projects with higher utilization ability.

The paper serves as a pointer to fellow academicians in incorporating a class project that ties together various disciplines of Electrical Engineering. Such a course accomplishes the ABET's objective of having a capstone course in the curriculum. The paper also discusses the horizontals leaning that takes place among students along with providing the students the environment where they could practice the designing of a complex system. This provides the students confidence and autonomy.

I. INTRODUCTION

Motor vehicles are fast, powerful, and controlled by easily distracted humans. Be it a fender bender or a serious accident, a collision in the parking lot or a pile-up on the highway, these multi-ton tools pose serious potential risk to themselves and to others. As technology advances, so too does the design of safety measures put in place to protect drivers and passengers in the event of a crash−or better yet, to prevent the crash altogether. Autonomous car designs dampen the effects of human error, but for these vehicles to act, they must undertake a difficult feat; that is, to detect and react to their surroundings while in motion.[2]

Complex AIs that utilize specially designed cameras and software could allow for vehicles to navigate roadways, but before even reaching that point in development, there is an imminent issue to address first: keeping the vehicle from harming itself by its physical surroundings. Street lights, speed limit signs, crosswalks, these are all important information pieces that drivers must read and understand at any given time, but before software is designed to read and comprehend this, it is equally useful to protect the car and driver from stationary telephone poles, ditches, and so on. Basic risk assessment isn't limited to public road ways though; vehicles for freight movement, construction, even immobility-assistance devices can utilize this technology for safer travel.

Defensive movement may not be the end all to driver safety technology, but it is a reasonable starting point. My project examines defensive travels at a small scale, observing that of a miniature, remote controlled car. The user acts as the driver and directs this small vehicle as desired, but the vehicle retains an opinion in the matter. It will not allow itself to move in a way otherwise perceived as dangerous. Instead, it takes control and won't act on the driver's commands so long as the driver directs the vehicle to harm's way. User “error” will not convey against the vehicle’s self-preservation; sensors along the front and center of the vehicle and procedure steps in the vehicle's programming will give it the tools it needs to drive safely.

Further, the process in which this project was created − through the summation of lessons from numerous differing discipline-focused classes prior − is explored so as to examine its relevance to fellow academicians regarding the integration of past lessons with required work-autonomy to instruct students how to more confidently and readily independently connect their past education to present problem-solving through practice, research, and experimentation.

II. SYSTEM DESCRIPTION

1. Overview by Block Diagram

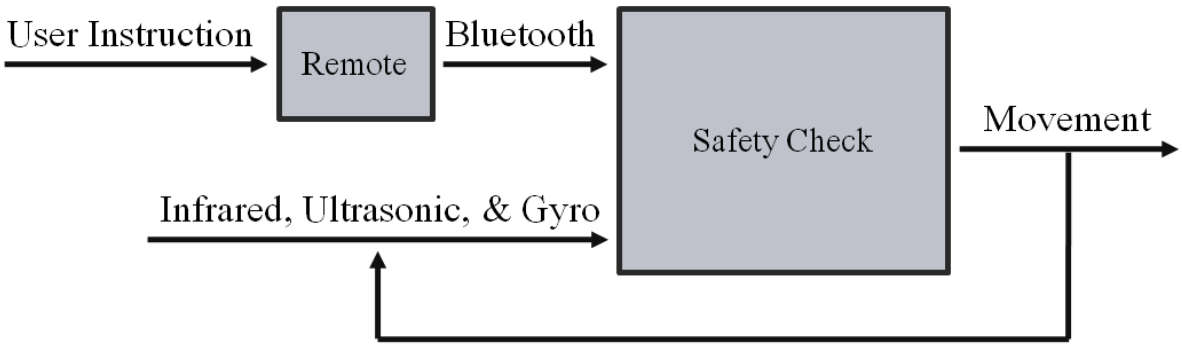


Fig. 1 Smart Miniature Vehicle Block Diagram

2. Vehicle Components and Data Collection

a. Boe-Bot Chassis (1)

Supplied by the Parallax education kit "Robotics with the Boe-Bot," this metal chassis served as the vehicle's main-body component. One large plastic wheel adorns either side and a small, sphere-shaped wheel is held by linchpin at its back. With numerous holes and slots in the metal panels, its design is very versatile and held the servo motors, both battery packs, and the constructed fiberglass platform.

b. Fiberglass Platform (1)

A drilled rectangle of fiberglass acts as the go-between that connects the microcontroller and sensors to the Boe-Bot chassis. Based after the high versatility of the Boe-Bot's metal chassis design, the platform contains eleven drilled holes, allowing objects to be screwed or tied into place securely. Metal spacers hold this platform 1.5 centimeters above the chassis, allowing for servo motor wires and a second battery pack to rest below, giving the vehicle area space to work around.

c. Servo Motor (2)

Also supplied by the Parallax education kit "Robotics with the Boe-Bot," both wheels are powered by Parallax Continuous Rotation Servo Motors. One battery pack (consisting of four AA batteries connected in series) is required to power both motors.

As depicted in "Fig.2 Servo Functions Code," the movement of the vehicle was developed around six requests. The "ONWARD," "BACKWARD," "RIGHTWARD," and "LEFTWARD" functions bring the robot in the direction their titles imply, while the calling of the function specified a unit of time−in microseconds−for the servo motors to act. The larger the time variable, the farther the vehicle moves while carrying out the function. For each servo, pulses are sent out throughout this time period and based on this second time variable, the servo may rotate in either direction. Around 1.5 ms theoretically leaves the servo with direction to turn, while greater or lesser amounts rotate the servo either clockwise or counterclockwise, giving the vehicle control over its direction of movement, turning ability, and speed.

However, leaving both servos set to not rotate won't necessarily keep the vehicle still as intended. Once on, the motors are always running, always reacting to each other, and rotating to some extent. As such, "GO_ON" and "STAY" functions are employed at the start of every movement request and throughout portions of the larger code as required. "GO_ON" activates the servo motors for use, whereas "STAY" de-activates them to keep the vehicle stationary. With a combination of these two functions called before and after the calling of movement functions, the servo motors' operations always remain predictable.

```
void ONWARD(int TIME)
{
  LEFT.writeMicroseconds(1625);         // Left -> CWW
  RIGHT.writeMicroseconds(1300);        // Right -> CW
  delay(TIME);
}

void BACKWARD(int TIME)
{
  LEFT.writeMicroseconds(1300);         // Left -> CW
  RIGHT.writeMicroseconds(1700);        // Right -> CW
  delay(TIME);
}

void RIGHTWARD(int TIME)
{
  LEFT.writeMicroseconds(1600);         // Left -> CCW
  RIGHT.writeMicroseconds(1600);        // Right -> CW
  delay(TIME);
}

void LEFTWARD(int TIME)
{
  LEFT.writeMicroseconds(1400);         // Left -> CW
  RIGHT.writeMicroseconds(1300);        // Right -> CWW
  delay(TIME);
}



void GO_ON()
{
  LEFT.attach(46);                  //Start servos (wheels now move)
  RIGHT.attach(44);                 //.
}

void STAY()
{
  LEFT.detach();                    // Stop servos (wheels stay still)
  RIGHT.detach();                   //.
}
```

Fig. 2 Servo Functions Code

d. Ultrasonic Sensor (2)

Empty floor readings−signifying potential gaps in the vehicle's path−were detected using two ultrasonic sensors (HC-SR04). One was placed directly in front of each wheel while pointed downward. Continuous centimeter-based readings are collected and compared with the designated 'standard' measurements. Where readings fall outside this narrow range, the vehicle is alerted to reverse back and then turn away from the detected problematic area. The ultrasonic sensors' detection-zone encompasses about 1.5 centimeters in front of the vehicle and about 1 centimeters to either side, giving reasonable warning time before the wheels may overcome a platform edge.

Depicted in "Fig. 3 Ultrasonic Readings", both the trigger and echo pins of the two sensors are designated and through a two-part process, the readings are taken and their units converted to centimeters. Later, in the "Safety Checklist" portion of the code, these readings are compared to the expected distance readings from the sensors to the ground and "BACKWARD" and "LEFTWARD"/"RIGHTWARD function calls initiated when necessary.

```
long Left_US_TRIG = 2;  long Left_US_ECHO = 3;      //Pins 2 & 3
long Right_US_TRIG = 5;    long Right_US_ECHO = 6;  //Pins 5 & 6

SR04 Left_sr04 = SR04(Left_US_ECHO,Left_US_TRIG);     //HC-SR04
SR04 Right_sr04 = SR04(Right_US_ECHO, Right_US_TRIG); //.

long Left_UltraSonic = Left_sr04.Distance();    //HC-SR04 distance values
long Right_UltraSonic = Right_sr04.Distance();  //.

Serial.print(Left_UltraSonic);  Serial.println("cm(L)");   ////TEST ULTRASONIC
Serial.print(Right_UltraSonic);  Serial.println("cm(R)");   ////TEST ULTRASONIC
Serial.println();
```

Fig. 3 Ultrasonic Readings

e. Infrared Sensor (3)

Infrared sensors (T1 3/4 IR LED and IR Receiver w/ 38kHz carrier frequency) detect potential obstacles the front of the vehicle may encounter. One infrared sensor is positioned at the above-corner of either side of the vehicle's front; these point forward and across−forming a quasi 'X' reading pattern−offering greater detection coverage than merely pointing the sensors forward would provide. To cover the shorter reading-point at the front-center of the vehicle, a third infrared sensor is added. Positioned below and behind the other two, it avoids hampering their line-of-site while it collects data for the remaining detection area.

Further sensor quantities were tested, but their use only provided diminishing returns. Whether due to the less than ideal remaining space for placement or perhaps due to the light pollution picked up from the main sensors, these additional sensors wantonly provided false-positive object pings at high rates. At its peak, seven infrared sensors (two

'X' reading patterns, one below the other; three center sensors forming a "triangle" pattern) were in simultaneous use with all programmed "object-detected" reactions requiring a minimum of two sensor confirmations. While the improved coverage would better protect the vehicle from otherwise hidden smaller obstacles, the false readings made this design impractical and unusable; the simpler, three sensor setup, was therefore instead devised.

When only one of either side's infrared sensor detects an object, the vehicle retreats and then turns away from the detected hazard. If the center sensor or both side sensors detect an object, the vehicle retreats only. The range of these sensors are each individually calibrated and are controlled by their frequency constants in the main program code.

To utilize infrared signals in this project, an output unit and receiver unit are required. "Fig. 4 Infrared Readings" shows the designating of their microcontroller pins, frequencies, and implementation. To ensure that the receivers both pick up the presence of potential objects and doesn't react to the very floor they are angled toward, the infrared frequencies have to be carefully calibrated through trial and error until their readings are to be considered reliable.

```
//Defining IR LED ad Receiver pins
int Left_Higher_IR_Led = 24; int Left_Higher_IR_Receiver = 22;       //
int Right_Higher_IR_Led = 28; int Right_Higher_IR_Receiver = 26;  //
int Middle_IR_Led = 32; int Middle_IR_Receiver = 30;                    //.

long Left_Higher_IR_Frequency = 43000;      // Higher Hz makes bot near-sighted
long Right_Higher_IR_Frequency = 43000;    //
long Middle_IR_Frequency = 45000;              //.

                //Square wave tone...  | time in ms...  | Read pin. 0 = item, 1 = no item...
tone(Left_Higher_IR_Led, Left_Higher_IR_Frequency, 8);             //
delay(1);                                                                  //
int Left_Higher_InfraRed = digitalRead(Left_Higher_IR_Receiver);       //
delay(3);                                                                  //
tone(Right_Higher_IR_Led, Right_Higher_IR_Frequency, 8);        //
delay(1);                                                                  //
int Right_Higher_InfraRed = digitalRead(Right_Higher_IR_Receiver);    //
delay(3);                                                                  //
tone(Middle_IR_Led, Middle_IR_Frequency, 8);                          //
delay(1);                                                                  //
int Middle_InfraRed = digitalRead(Middle_IR_Receiver);                     //
delay(3);                                                                  //.
```

Fig. 4 Infrared Readings

f. Gyroscope (1)

Placed in the center cavity of the microcontroller and below much of the wiring, lays the gyroscope (GY-521 Module). Setup to designate it's horizontal position as 90° and any tilting as a value between 0° and 90°, it will command the vehicle to retreat a distance if any angle below 60° is measured. As the vehicle is top heavy and only utilized two main wheels, this was determined as an adequate alarm point to call for self-action.

"Fig. 5 Gyroscopic Readings" depicts some of the necessitated code required to make measurements with the gyroscope. The minimum and maximum variable values are dependent on Arduino unit, but as the gyroscope detected orientation very accurately in initial testing, it was decided to leave these two values as were. After this, each axis of the gyro provides its own measurements, which are converted and updated into degree units. Here their values were adjusted and then further calibrated so as to designate the gyroscope's horizontal placement as 90° as was found to be the easiest base number to reference further tilt tests against.

```
int MinimumValue=265;
int MaximumValue=402;

double Gyro_X;  double Gyro_Y;
Gyro_Accel_X = Wire.read() << 8|Wire.read();  //
Gyro_Accel_Y = Wire.read() << 8|Wire.read();  //
Gyro_Accel_Z = Wire.read() << 8|Wire.read();  //.

//MPU data
int Gyro_Angle_X =-map(Gyro_Accel_X, MinimumValue, MaximumValue, -90, 90);
int Gyro_Angle_Y = map(Gyro_Accel_Y, MinimumValue, MaximumValue, -90, 90);
int Gyro_Angle_Z = map(Gyro_Accel_Z, MinimumValue, MaximumValue, -90, 90);

//Give positive values at 90 deg.
Gyro_X= RAD_TO_DEG * (atan2(-Gyro_Angle_Y, -Gyro_Angle_Z)+PI)-270;//
Gyro_Y= RAD_TO_DEG * (atan2(-Gyro_Angle_X, -Gyro_Angle_Z)+PI)-270; //.

//Convert negative values to respective positive 0-90 deg.
if (Gyro_X < 0) {Gyro_X = -Gyro_X - 180;}
if (Gyro_Y < 0) {Gyro_Y = -Gyro_Y - 180;}
```

Fig. 5 Gyroscopic Readings

g. Bluetooth Module (1)

The Bluetooth (HC-05 Module) module connects to the microcontroller and receives data inputs from an Android Bluetooth app, "Arduino Bluetooth Control" by Broxcode.[6] Each direction is simplified to a letter; 'G' for "Move forward" ("Go"), 'L' for "Turn Left," 'R' for "Turn Right," and 'S' for "Stop movement," which is sent to the microcontroller to be read by its serial monitor. From there the listing passes through

loops within the code and, if no sensors detect an external "threat," the vehicle acts on these signals and moves accordingly.

Directly in the "Safety Checklist," resides the code lines directing the vehicle to move are requested when deemed safe; depicted in "Fig. 6 Bluetooth Communications." Also depicted are examples of call functions. While the rest of the safety checklist may utilize multiple requests per instance (e.g., "GO_ON(); BACKWARD(1200); STAY(); delay(200); GO_ON(); LEFTWARD(375);" to bring the vehicle away from a ledge.), the motion requests need not be complex nor hold extended runtimes. Where no potential harm is detected, these function calls loop as needed until the user provides a new requested motion direction.

```
else if(Serial.available())
    {
     while(Serial.available())
     {
      char In_Direction = (char)Serial.read(); //read the input data
      In_String += In_Direction;       //Incoming serial character forms a 'string'
     }

     Serial.println(In_String);

     while (Serial.available() > 0)
     {
      CONTROLLER = Serial.read() ;
     }     // clear the serial buffer

     if(In_String == "G")       //Go forward//
     {           GO_ON(); ONWARD(500);       }

     else if(In_String == "L")   //Head Left//
     {         GO_ON(); LEFTWARD(400);       }

     else if(In_String == "R")   //Head Right//
     {         GO_ON(); RIGHTWARD(300);       }

     else if(In_String == "S")   //Stop movement//
     {         STAY();       }

     In_String = "";
    }
```

Fig. 6 Bluetooth Communications

h. Microcontroller (1)

The microcontroller (Arduino Mega 2560) process all incoming sensor signals and runs the program that directs the vehicle, by way of servo motors, to move and turn. It is powered by one battery pack (also containing 4 AA batteries connected in series) attached below the metal chassis and is screwed directly to the fiberglass platform. With four VCC pins (three 5V and one 3.5V) and five GND pins, the microcontroller transfers power from the battery pack to all the vehicles sensors−ultrasonic, infrared, gyroscope, and Bluetooth. It is through this microcontroller that the code is uploaded and runs. (See Appendix 2f for full program code.)

3. Vehicle Movement Signals and Reactions

As the vehicle only holds sensors on its front-most side, all "threat" detection, besides gyroscope readings, is based on only front-most events. With a short detection range due to the limitations of the sensors provided, where a forthcoming object or gap in the path is detected, the vehicle instantly must react. If the point of concern is found to be on either side of the vehicle, the vehicle retreats a distance and turns away from the supposed threat. If the point of concern is only detected straight on, the vehicle merely backs away a great enough distance so as to assume the controller may turn either way to dodge the item themselves.

While the controller may continuously attempt to re-drive the vehicle to the item of determined harm, the vehicle can only retreat time and again as it has no external reason to direct the user to which side, left or right, to travel; with no knowledge to the desired path and no ability to look farther than its sensors allow, the vehicle does not exceed its directive by acting in anyway besides self-preservation; the user always maintains total creative-control in the vehicle's path choices.

III. RESULTS AND FURTHER DISCUSSION

1. Vehicle Movement Findings

The vehicle acts within acceptable movement range while further design strategies are feasible to better serve greater intended goals if it were to be re-attempted in a later class. Through much experimentation and many re-designs, every individual component of this vehicle justifiably works as programmed.

As through calculation, movement forward stays within approximately 5° of that of a straight path for every eight feet of forward distance while backward movement stays within approximately 10° for every eight feet of backward distance. Detection of drops occur readily within 2.5" of a platform edge while obstacle detection ranges from 2" to 6" at the vehicles front. This variance was observed to follow the ability of the individual infrared sensors and did not necessarily depend on sensor placement in conjunction to the vehicle. Items below 0.5" more readily avoid sensor detection until the vehicle moves to within approximately 2" of the object.

In summary, the infrared and ultrasonic sensors detect object presence and distance accurately and alert the vehicle in a quick manner, giving the microcontroller time to decide and act. Large items are readily detectable, and even minor drops in path are noted accordingly. The vehicle acts well exploring rooms and large pieces of furniture, and the sensors stay well placed on the vehicle regardless of how it is spun, sent forward, or suddenly stopped. The gyroscope can read tilt angle in exact degrees and alerts when high tilt is detected either on its X or Y axis. Placed on a board and tilted to relatively exact measurements, the vehicle reacts accurately and behaves just and is requested in its program; where horizontal is considered 90°, at 60° the vehicle retreats just as designed. Lastly, the Bluetooth module communicates commands between the smart phone and the microcontroller well and allows for communication across a classroom's length reliably. The vehicle drive straight forward with little curve, turns slowly when requested, and stops on command. Individually and in controlled tests, these units and their written code work just and exactly as planned.

Yet, in conjunction, these elements are not as synchronous as desired when the vehicle is in motion. These anomalies have been analyzed thoroughly over many weeks, and every attempt to better correct available readings has been made, but the potential false-positives continue to appear, leading to the suspicion that it is not the design or build, but the sensor units' quality itself, that is to be called into question.

The gyroscope has been found to react to an unknown element and gives alarms where no tilt-danger is present, but this reaction only occurs when the servo motors bring movement to the vehicle. Transporting the vehicle quickly by hand doesn't set off this trigger, leading to a hard to recreate symptom. As the servo motors are powered by a separate battery unit, it is unlikely that they are the direct cause of fault, but no other repeatable pattern has yet been detected. Other quandaries of interest are simple occasional false-positives from the ultrasonic sensors or infrared sensors that sometimes occur. Also, most Bluetooth connection apps are free, but also prone to occasionally freezing; when this spontaneous event occurs, the vehicle still maintains its sensors and thus will still react to upcoming dangers, but it is also now on its own until the app can be brought back up and reconnected to the vehicle's Bluetooth module. It's not a major issue in light of the sensor testings, but is still an area of

further research as the user is meant to maintain full control where no danger is present while this hindrance exclaims otherwise.

2. Project Improvements by a Separate Senior Design Class or Personal Further Study

Much of the sensors' false-positive readings may be discernibly attributed to the low quality of the sensors used. With design strategies fluid and no previous experience regarding utilizing all sensor elements in unison in a fully completed build, desired project sensors were purposely limited to those readily available and low in cost. This allowed greater flexibility in design and re-design, but foreseeable improvements are now fully reportable; a re-focus on the entire project, using the design elements learned from this implementation, could potentially provide a strong basis for a more accurate, more stream-lined "smart miniature vehicle."

Perhaps with this trial-and-error process complete, further sensor implantation could provide a better image to the vehicle regarding its surrounding hazards. Large items are detectable, but smaller items can still come close to the vehicle. Previous investigation found that greater numbers of sensors merely polluted the testing fields and thus gave little improvement to data collection, but if higher quality sensors were used, perhaps they could then be placed in closer proximity to one another.

While undetectable until the gyroscope acted in unison with the servo motors, the potential gyroscope misreadings could potentially be counteracted with a second sensor unit. Research into the matter brought up experiments where two gyroscope units are used and their data combined for a more reliable reading. Prior this was noted, but not followed up on as the current tests produced by the lone gyroscope were deemed satisfactory. Yet, in light on the final gyroscope readings while the servo motors are active, this research would be notable to follow-up on in a further study.

Lastly, there are Bluetooth controller method could explore further research paths and take benefit from potential improvement if a later experiment were to be conducted. While research was collected and parts assembled for a fully separate controller that could be run off of a second Arduino unit and a second Bluetooth module, this plan was redesigned mid-way through its build. Research depicted a modification to the HC-06 Bluetooth unit that required exceedingly precise soldering skill. As soldering was a skill I am still new too, it was feared that the Bluetooth chip would become irreparably damaged and thus the external controller build was halted and the smart phone app was researched instead. As this secondary method relies on a third-party software, if the two Bluetooth module plan was not utilized in a follow-up project, research into a custom app could be explored instead.

IV. INCORPORATION TO FURTHER EDUCTATION

1. Combination of Prior Education to form a Multi-Encompassing Project

Design elements and methods to complete this project encompass teachings from classes the student already took prior and include, but were not limited to:

ECET 15201 − AC Circuits And Analysis
ECET 15401 − Electronic Components Circuits
ECET 15901 − Digital Circuits And Applications
ECET 20901 − Microcontroller Applications
ECET 21201 − Electrical Power And Motors
ECET 31201 − Power Electronics Fundamentals
ECET 40400 − Wireless Communication And Networking
ECET 45500 − Object Oriented System Design
ECET 45600 − Operating System With Embedded System Design
IET 30800 − Engineering Project Management And Economic Analysis

Students are encouraged to research and select their own project topic with a given guideline to focus the project further than the scope of prior class assignments. The students are expected to choose project ideas that may use elements similar to those they are familiar with, but the project also must utilizes components, theories, and tasks that are brand new. This forces the student to research and experiment with new concepts while they simultaneously practice and perfect the tasks they are already familiar with. Lessons they may not have reviewed since the beginning of their higher education suddenly may require re-study and concepts learned recently may now better tie back to their past fundamentals as research and experimentation progresses.

In the example of this present project, the basic concepts of circuit layout, C++ programming, and project management theory had been covered prior, but the practical use of numerous different senor components simultaneously was new as was the act of soldering, the initial coding in PBASIC, Bluetooth communication, and the essential process of carrying out a long-term project with efficient time management. Independent research regarding aspects not prior covered in class forced the student to proactively collect and study new material and self-determine the sustainability of new design ideas as the project progressed.

Through combining the various disciplines of Electrical Engineering into a single class project, the ABET's objective regarding the inclusion of a capstone course in the curriculum is achieved. As the students work to complete their chosen project, they gain a level of confidence and autonomy not formally provided in their previous education experiences. This multi-encompassing projects obliges them to work through setbacks that before they could dismiss as just "a bad assignment;" in this procedure, they were the student who chose the topic, and they are the student that must manage long-term deadlines regarding a project that encompasses past learned theories and skills. With the student's past knowledge and ability at the forefront of the assignment, they are forced to work independently and expand their abilities to levels more fitting of their future goals.

V. CONCLUSION

Utilizing ultrasonic and infrared sensors, a small vehicle was built and programmed to appropriately detect oncoming objects and changes in its distance relative to the ground. These sensors served to provide the vehicle a means to avoid driving into obstacles or off upcoming ledges. Positioning a gyroscope at its center, the vehicle detects changes in tilt angle and instructs the vehicle to retreat accordingly to a safer landing. To collect this data and direct required counter-actions, an Arduino Mega microcontroller was used. The two servo motors that powered the vehicle's wheels required a greater power output than the microcontroller could redirect from its connected battery pack composed of four AA batteries run in series; therefore, a second battery pack was implemented for sole servo motor use. Moreover, a Bluetooth module connected to the vehicle allowed for external direction commands to the vehicle. Letters were assigned to arrows in a special Bluetooth app and this data was sent to the vehicle where if would activate functions commanding the vehicle to either move straight, turn left, turn right, or halt movement.

Overall, the small vehicle did well moving under user provided direction commands while its four sensor groups−infrared, ultrasonic, and gyroscope−monitored its movement for potential dangers and controlled the vehicle as necessary to maintain safe navigation. However, due to the limitations of the low-priced sensors, some errors in data collection were found as the vehicle travels. The most notable error was the false-positive obtained periodically by the gyroscope when used with the servo motors. While not a constant error, it's misguided alerts to the main program can slow the forward progress of the vehicle's motions. Further, the Bluetooth controller relies on a third party app, but, it is believed that this program could potentially be written by a student to more fully provide a Purdue-learning based tool to even this aspect of the project. Or, more solder-experienced of students could complete the challenging step of soldering a very small pin of a second Bluetooth module, and then it would be ready for a second Arduino-based Bluetooth controller. Lastly, with the overall design and implementation work completed, more expensive−and exact−sensors could be obtained and implemented to provide better readings and perhaps even now allow for a higher quantity of sensors placed to give the vehicle a clearer picture of its surroundings.

This Senior Design II project succeeded in its initiative to create a Bluetooth controlled vehicle that utilized sensors to detect, and interact, with its surroundings. With its gathered data, new jumping-off points have been revealed that may fuel further student experimentation and research in the subject of sensor-based vehicle navigation. Where phase I explored the already mapped project utilizing Parallax's Boe-Bot kit, provided supplies, and instruction booklet, and phase II explored self implemented sensors, programming, and build-designs, a potential phase III project has room to explore higher powered, more reliable sensors, student created Bluetooth instruments, and further testing regarding the comprehensive, multi-sensor system that is the "smart miniature vehicle."

Through the necessary use of a varied set of prior knowledge and new skills and research, this student project serves as an extended learning assignment that better prepared the student for further in-depth learning exercises. Distinct connections are made to prior classes' learning objectives while a strong emphasis is given to independent research and exploration of new technologies and methods as equired for the project's completion.

VI. BIBLIOGRAPHY

[1] Inc, P. (Version 3.0). *Robotics with the Boe-Bot: Student Guide* .

[2] *Automated Vehicles for Safety*. (2018, April 20). Retrieved from National Highway Traffic Safety Administration: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[3] *Continuous Rotatio Servo Documentation.* (2012, September 20). Retrieved November 3, 2018, from parallax.com: https://www.parallax.com/sites/default/files/downloads/900-00008-Continuous-Rotation-Servo-Documentation-v2.2.pdf

[4] Elegoo. (V1.0.17.4.7). *ELEGOO: THE MOST COMPLETE STARTER KIT TUTORIAL FOR MEGA2560.* Retrieved from EUservice@elegoo.com.

[5] *5mm Infrared LED , T-1 3/4.* (2016, March 23). Retrieved November 5, 2018, from cdn-shop.adafruit.com: https://cdn-shop.adafruit.com/datasheets/IR333_A_datasheet.pdf

[6] *IR Receiver Modules for Remote Control Systems.* (2018, November 2). Retrieved November 3, 2018, from vishay.com: https://www.vishay.com/docs/82491/tsop382.pdf

[7] *MPU-6000 and MPU-6050 PRoduct Specification.* (2017, September 27). Retrieved March 6, 2018, from store.invensense.com: https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf

[8] *HC-05 - Bluetooth to Serial Module*. (2018, August 20). Retrieved November 1, 2018, from electronicaestudio.com: http://www.electronicaestudio.com/docs/istd016A.pdf

[9] *ATmega2560-Arduino Pin Mapping*. (n.d.). Retrieved November 29, 2018, from arduino.cc: https://www.arduino.cc/en/Hacking/PinMapping2560

[10] Lindsay, A. (Version 1.1). *Robotics with the BOE Shield-Bot for Arduino.*

[11] Kirnik, B. (n.d.). *How to Measure Angle With MPU-6050(GY-521).* (n.d.). Retrieved October 29, 2018, from instructables.com: https://www.instructables.com/id/How-to-Measure-Angle-With-MPU-6050GY-521/

[12] Tariq, H. (n.d.). *Remote Controlled LED Using HC-05 Bluetooth, Arduino and Mobile Phone App*. Retrieved November 9, 2018, from instructables.com: https://www.instructables.com/id/Remotely-Control-LED-using-HC-05-Bluetooth-Arduino/

[13] *Download*. (n.d.). Retrieved 12 1, 2018, from fritzing.org: http://fritzing.org/download/

VII. APPENDICIES

1. Data Sheet

a. Parallax Continuous Rotation Servo Motors [3]

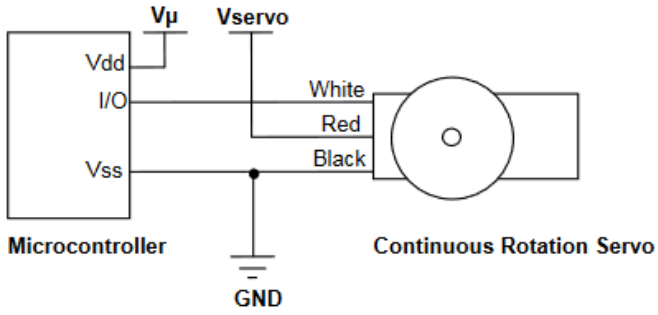
Features

- Bidirectional continuous rotation
- 0 to 50 RPM, with a linear response to PWM for easy ramping
- Accepts four mounting screws
- Easy to interface with any Parallax microcontroller or PWM-capable device
- Very easy to control with the PULSOUT command in PBASIC or SX/B
- Weighs only 1.50 oz (42.5 g)
- 38 oz-in torque @ 6 V

Key Specifications

- Power requirements: 4 to 6 VDC; Maximum current draw 140 +/- 50 mA at 6 VDC when operating in no load conditions, 15 mA when in static state
- Communication: pulse-width modulation
- Dimensions: approx 2.2 x 0.8 x 1.6 in (5.58x 1.9 x 4.06 cm) excluding servo horn
- Operating temperature range: 14 to 122 °F (-10 to +50 °C)

Quick-Start Circuit



Vμ = microcontroller voltage supply

Vservo = 4 to 6 VDC, regulated or battery

I/O = PWM TTL or CMOS output signal, 3.3 to 5 V; < Vservo + 0.2 V

Specifications

Pin	Name	Description	Minimum	Typical	Maximum	Units
1 (White)	Signal	Input; TTL or CMOS	3.3	5.0	Vservo + 0.2	V
2 (Red)	Vservo	Power Supply	4.0	5.0	6.0*	V
3 (Black)	Vss	Ground		0		V

b. HC-SR04 [4]

Component Introduction

Ultrasonic sensor

Ultrasonic sensor module HC-SR04 provides 2cm-400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

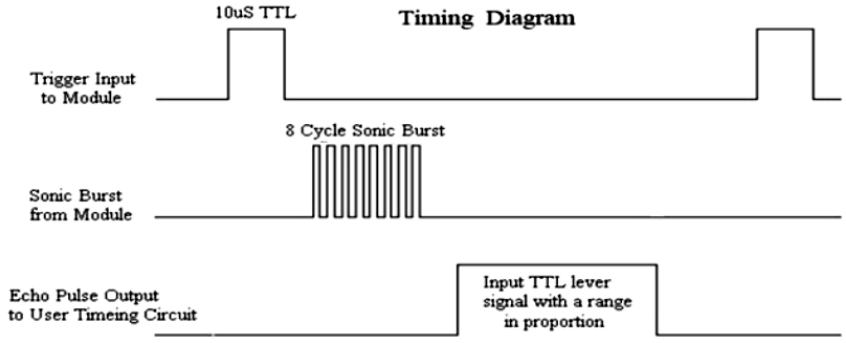
(1) Using IO trigger for at least 10us high level signal,

(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.

(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic tore turning.

Test distance = (high level time × velocity of sound (340m/s) /2

The Timing diagram is shown below. You only need to supply a short 10us pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: us / 58 = centimeters or us / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



c. T1 3/4 IR LED [5]

Absolute Maximum Ratings (Ta=25℃)

Parameter	Symbol	Rating	Units
Continuous Forward Current	I_F	100	mA
Peak Forward Current	I_{FP}	1.0	A
Reverse Voltage	V_R	5	V
Operating Temperature	T_{opr}	-40 ~ +85	℃
Storage Temperature	T_{stg}	-40 ~ +85	℃
Soldering Temperature	T_{sol}	260	℃
Power Dissipation at(or below) 25℃Free Air Temperature	P_d	150	mW

Notes: *1:I_{FP} Conditions--Pulse Width≦100 μs and Duty≦1%.

*2:Soldering time≦5 seconds.

Electro-Optical Characteristics (Ta=25℃)

Parameter	Symbol	Condition	Min.	Typ.	Max.	Units
Radiant Intensity	Ee	I_F=20mA	7.8	20	--	mW/sr
		I_F=100mA Pulse Width≦100 μs ,Duty≦1%	--	85	--	
		I_F=1A Pulse Width≦100 μs ,Duty≦1%.	--	750	--	
Peak Wavelength	λp	I_F=20mA	--	940	--	nm
Spectral Bandwidth	Δλ	I_F=20mA	--	45	--	nm
Forward Voltage	V_F	I_F=20mA		1.2	1.5	V
		I_F=100mA Pulse Width≦100 μs ,Duty≦1%	--	1.4	1.8	
		I_F=1A Pulse Width≦100 μs ,Duty≦1%.	--	2.6	4.0	
Reverse Current	I_R	V_R=5V	--	--	10	μA
View Angle	2θ1/2	I_F=20mA	--	20	--	deg

d. IR Receiver w/ 38kHz Carrier Frequency [6]

ELECTRICAL AND OPTICAL CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)						
PARAMETER	**TEST CONDITION**	**SYMBOL**	**MIN.**	**TYP.**	**MAX.**	**UNIT**
Supply current	E_v = 0, V_S = 3.3 V	I_{SD}	0.27	0.35	0.45	mA
	E_v = 40 klx, sunlight	I_{SH}	-	0.45	-	mA
Supply voltage		V_S	2.5	-	5.5	V
Transmission distance	E_v = 0, test signal see Fig. 1, IR diode TSAL6200, I_F = 50 mA	d	-	24	-	m
Output voltage low	I_{OSL} = 0.5 mA, E_e = 0.7 mW/m², test signal see Fig. 1	V_{OSL}	-	-	100	mV
Minimum irradiance	Pulse width tolerance: $t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$, test signal see Fig. 1	$E_{e\,min.}$	-	0.12	0.25	mW/m²
Maximum irradiance	$t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o$, test signal see Fig. 1	$E_{e\,max.}$	30	-	-	W/m²
Directivity	Angle of half transmission distance	$\varphi_{1/2}$	-	± 45	-	°

TYPICAL CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)

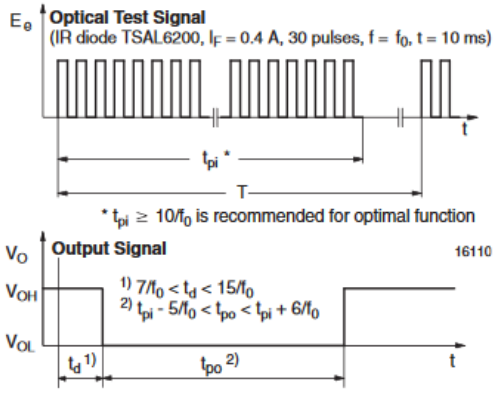


Fig. 1 - Output Active Low

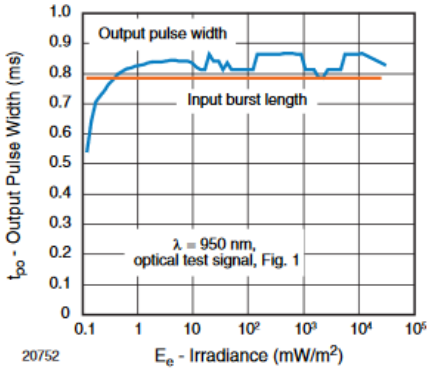


Fig. 2 - Pulse Length and Sensitivity in Dark Ambient

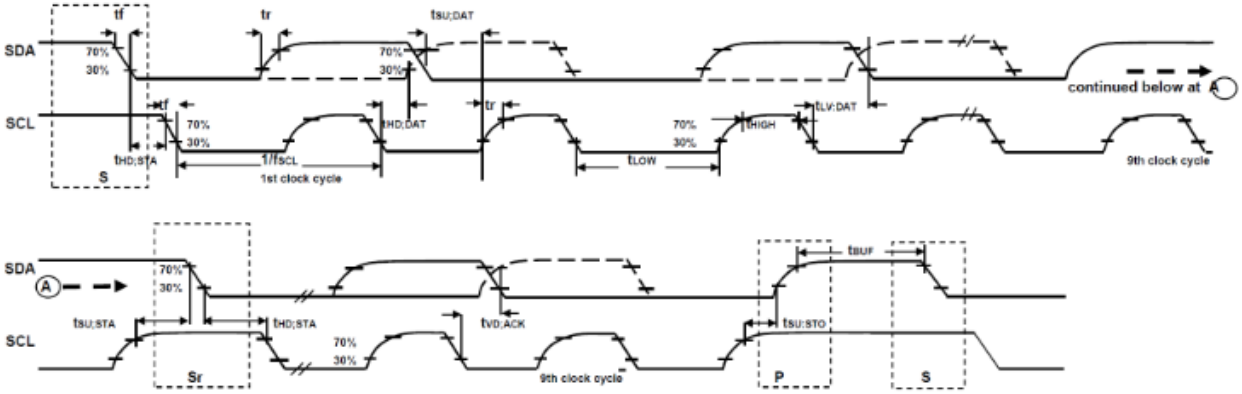
e. GY-521 Module [7]

6.7 I²C Timing Characterization

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Min	Typical	Max	Units	Notes
I²C TIMING	**I²C FAST-MODE**					
f_{SCL}, SCL Clock Frequency				400	kHz	
$t_{HD.STA}$, (Repeated) START Condition Hold Time		0.6			µs	
t_{LOW}, SCL Low Period		1.3			µs	
t_{HIGH}, SCL High Period		0.6			µs	
$t_{SU.STA}$, Repeated START Condition Setup Time		0.6			µs	
$t_{HD.DAT}$, SDA Data Hold Time		0			µs	
$t_{SU.DAT}$, SDA Data Setup Time		100			ns	
t_r, SDA and SCL Rise Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
t_f, SDA and SCL Fall Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
$t_{SU.STO}$, STOP Condition Setup Time		0.6			µs	
t_{BUF}, Bus Free Time Between STOP and START Condition		1.3			µs	
C_b, Capacitive Load for each Bus Line			< 400		pF	
$t_{VD.DAT}$, Data Valid Time				0.9	µs	
$t_{VD.ACK}$, Data Valid Acknowledge Time				0.9	µs	

Note: Timing Characteristics apply to both Primary and Auxiliary I²C Bus



I²C Bus Timing Diagram

f. HC-05 [8]

PIN Name	PIN #	Pad type	Description	Note
GND	13 21 22	VSS	Ground pot	
3.3 VCC	12	3.3V	Integrated 3.3V (+) supply with On-chip linear regulator output within 3.15-3.3V	
AIO0	9	Bi-Directional	Programmable input/output line	
AIO1	10	Bi-Directional	Programmable input/output line	
PIO0	23	Bi-Directional RX EN	Programmable input/output line, control output for LNA(if fitted)	
PIO1	24	Bi-Directional TX EN	Programmable input/output line, control output for PA(if fitted)	

PIO2	25	Bi-Directional	Programmable input/output line	
PIO3	26	Bi-Directional	Programmable input/output line	
PIO4	27	Bi-Directional	Programmable input/output line	
PIO5	28	Bi-Directional	Programmable input/output line	
PIO6	29	Bi-Directional	Programmable input/output line	
PIO7	30	Bi-Directional	Programmable input/output line	
PIO8	31	Bi-Directional	Programmable input/output line	
PIO9	32	Bi-Directional	Programmable input/output line	
PIO10	33	Bi-Directional	Programmable input/output line	
PIO11	34	Bi-Directional	Programmable input/output line	

g. Arduino Mega 2560 [9]

Pin Number	Pin Name	Mapped Pin Name
1	PG5 (OC0B)	Digital pin 4 (PWM)
2	PE0 (RXD0/PCINT8)	Digital pin 0 (RX0)
3	PE1 (TXD0)	Digital pin 1 (TX0)
4	PE2 (XCK0/AIN0)	
5	PE3 (OC3A/AIN1)	Digital pin 5 (PWM)
6	PE4 (OC3B/INT4)	Digital pin 2 (PWM)
7	PE5 (OC3C/INT5)	Digital pin 3 (PWM)
8	PE6 (T3/INT6)	
9	PE7 (CLKO/ICP3/INT7)	
10	VCC	VCC
11	GND	GND
12	PH0 (RXD2)	Digital pin 17 (RX2)
13	PH1 (TXD2)	Digital pin 16 (TX2)
14	PH2 (XCK2)	
15	PH3 (OC4A)	Digital pin 6 (PWM)
16	PH4 (OC4B)	Digital pin 7 (PWM)
17	PH5 (OC4C)	Digital pin 8 (PWM)
18	PH6 (OC2B)	Digital pin 9 (PWM)
19	PB0 (SS/PCINT0)	Digital pin 53 (SS)
20	PB1 (SCK/PCINT1)	Digital pin 52 (SCK)
21	PB2 (MOSI/PCINT2)	Digital pin 51 (MOSI)
22	PB3 (MISO/PCINT3)	Digital pin 50 (MISO)
23	PB4 (OC2A/PCINT4)	Digital pin 10 (PWM)
24	PB5 (OC1A/PCINT5)	Digital pin 11 (PWM)
25	PB6 (OC1B/PCINT6)	Digital pin 12 (PWM)
26	PB7 (OC0A/OC1C/PCINT7)	Digital pin 13 (PWM)
27	PH7 (T4)	
28	PG3 (TOSC2)	
29	PG4 (TOSC1)	
30	RESET	RESET
31	VCC	VCC

32	GND	GND
33	XTAL2	XTAL2
34	XTAL1	XTAL1
35	PL0 (ICP4)	Digital pin 49
36	PL1 (ICP5)	Digital pin 48
37	PL2 (T5)	Digital pin 47
38	PL3 (OC5A)	Digital pin 46 (PWM)
39	PL4 (OC5B)	Digital pin 45 (PWM)
40	PL5 (OC5C)	Digital pin 44 (PWM)
41	PL6	Digital pin 43
42	PL7	Digital pin 42
43	PD0 (SCL/INT0)	Digital pin 21 (SCL)
44	PD1 (SDA/INT1)	Digital pin 20 (SDA)
45	PD2 (RXD1/INT2)	Digital pin 19 (RX1)
46	PD3 (TXD1/INT3)	Digital pin 18 (TX1)
47	PD4 (ICP1)	
48	PD5 (XCK1)	
49	PD6 (T1)	
50	PD7 (T0)	Digital pin 38
51	PG0 (WR)	Digital pin 41
52	PG1 (RD)	Digital pin 40
53	PC0 (A8)	Digital pin 37
54	PC1 (A9)	Digital pin 36
55	PC2 (A10)	Digital pin 35
56	PC3 (A11)	Digital pin 34
57	PC4 (A12)	Digital pin 33
58	PC5 (A13)	Digital pin 32
59	PC6 (A14)	Digital pin 31
60	PC7 (A15)	Digital pin 30
61	VCC	VCC
62	GND	GND
63	PJ0 (RXD3/PCINT9)	Digital pin 15 (RX3)
64	PJ1 (TXD3/PCINT10)	Digital pin 14 (TX3)
65	PJ2 (XCK3/PCINT11)	
66	PJ3 (PCINT12)	
67	PJ4 (PCINT13)	
68	PJ5 (PCINT14)	
69	PJ6 (PCINT 15)	
70	PG2 (ALE)	Digital pin 39
71	PA7 (AD7)	Digital pin 29
72	PA6 (AD6)	Digital pin 28
73	PA5 (AD5)	Digital pin 27

74	PA4 (AD4)	Digital pin 26
75	PA3 (AD3)	Digital pin 25
76	PA2 (AD2)	Digital pin 24
77	PA1 (AD1)	Digital pin 23
78	PA0 (AD0)	Digital pin 22
79	PJ7	
80	VCC	VCC
81	GND	GND
82	PK7 (ADC15/PCINT23)	Analog pin 15
83	PK6 (ADC14/PCINT22)	Analog pin 14
84	PK5 (ADC13/PCINT21)	Analog pin 13
85	PK4 (ADC12/PCINT20)	Analog pin 12
86	PK3 (ADC11/PCINT19)	Analog pin 11
87	PK2 (ADC10/PCINT18)	Analog pin 10
88	PK1 (ADC9/PCINT17)	Analog pin 9
89	PK0 (ADC8/PCINT16)	Analog pin 8
90	PF7 (ADC7)	Analog pin 7
91	PF6 (ADC6)	Analog pin 6
92	PF5 (ADC5/TMS)	Analog pin 5
93	PF4 (ADC4/TMK)	Analog pin 4
94	PF3 (ADC3)	Analog pin 3
95	PF2 (ADC2)	Analog pin 2
96	PF1 (ADC1)	Analog pin 1
97	PF0 (ADC0)	Analog pin 0
98	AREF	Analog Reference
99	GND	GND
100	AVCC	VCC

2. Program Code

a. Servo Motor Code Reference [10]

```
/*
Robotics with the BOE Shield - ServosOppositeDirections
Generate a servo full speed counterclockwise signal with pin 13 and
full speed clockwise signal with pin 12.
*/

#include <Servo.h>                      // Include servo library

Servo servoLeft;                        // Declare left servo signal
Servo servoRight;                       // Declare right servo signal

void setup()                            // Built in initialization block
{
  servoLeft.attach(13);                 // Attach left signal to pin 13
  servoRight.attach(12);                // Attach right signal to pin 12

  servoLeft.writeMicroseconds(1700);    // 1.7 ms -> counterclockwise
  servoRight.writeMicroseconds(1300);   // 1.3 ms -> clockwise
}

void loop()                             // Main loop auto-repeats
{                                       // Empty, nothing needs repeating
}
```

b. HC-SR04 Code Reference [4]

```
//www.elegoo.com
//2016.12.08
#include "SR04.h"
#define TRIG_PIN 12
#define ECHO_PIN 11
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);
long a;

void setup() {
   Serial.begin(9600);
   delay(1000);
}

void loop() {
   a=sr04.Distance();
   Serial.print(a);
   Serial.println("cm");
   delay(1000);
}
```

c. IR Sensor Code Reference [10]

```
/*
 * Robotics with the BOE Shield - TestLeftIR
 * Display 1 if the left IR detector does not detect an object,
 * or 0 if it does.
 */

void setup()                                // Built-in initialization block
{
  tone(4, 3000, 1000);                      // Play tone for 1 second
  delay(1000);                              // Delay to finish tone

  pinMode(10, INPUT);  pinMode(9, OUTPUT);   // Left IR LED & Receiver

  Serial.begin(9600);                       // Set data rate to 9600 bps
}

void loop()                                 // Main loop auto-repeats
{
  int irLeft = irDetect(9, 10, 38000);  // Check for object

  Serial.println(irLeft);                   // Display 1/0 no detect/detect

  delay(100);                               // 0.1 second delay
}

// IR Object Detection Function

int irDetect(int irLedPin, int irReceiverPin, long frequency)
{
  tone(irLedPin, frequency, 8);             // IRLED 38 kHz at least 1 ms
  delay(1);                                 // Wait 1 ms
  int ir = digitalRead(irReceiverPin);  // IR receiver -> ir variable
  delay(1);                                 // Down time before recheck
  return ir;                                // Return 1 no detect, 0 detect
}
```

d. GY-521 Module Code Reference [11]

```
//Written by Ahmet Burkay KIRNIK
//Measurement of Angle with MPU-6050(GY-521)

#include<Wire.h>

const int MPU_addr=0x68; int16_t
AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;

int minVal=265; int maxVal=402;

double x; double y; double z;

void setup(){ Wire.begin();
Wire.beginTransmission(MPU_addr);
Wire.write(0x6B); Wire.write(0);
Wire.endTransmission(true); Serial.begin(9600); } void
loop(){ Wire.beginTransmission(MPU_addr);
Wire.write(0x3B); Wire.endTransmission(false);
Wire.requestFrom(MPU_addr,14,true);
AcX=Wire.read()<<8|Wire.read(); AcY=Wire.read()
<<8|Wire.read(); AcZ=Wire.read()<<8|Wire.read(); int
xAng = map(AcX,minVal,maxVal,-90,90); int yAng =
map(AcY,minVal,maxVal,-90,90); int zAng =
map(AcZ,minVal,maxVal,-90,90);

x= RAD_TO_DEG * (atan2(-yAng, -zAng)+PI); y=
RAD_TO_DEG * (atan2(-xAng, -zAng)+PI); z=
RAD_TO_DEG * (atan2(-yAng, -xAng)+PI);

Serial.print("AngleX= "); Serial.println(x);

Serial.print("AngleY= "); Serial.println(y);

Serial.print("AngleZ= "); Serial.println(z);
Serial.println("-----------------------------------------
"); delay(400); }
```

e. HC-05Code Reference [12]

```
/*
Arduino Turn LED On/Off using Serial Commands
Created April 22, 2015
Hammad Tariq, Incubator (Pakistan)

It's a simple sketch which waits for a character on serial
and in case of a desirable character, it turns an LED on/off.

Possible string values:
a (to turn the LED on)
b (tor turn the LED off)
*/

char junk;
String inputString="";

void setup()                    // run once, when the sketch starts
{
 Serial.begin(9600);            // set the baud rate to 9600, same should be of your Serial Monit
or
 pinMode(13, OUTPUT);
}

void loop()


{
  if(Serial.available()){
  while(Serial.available())
    {
      char inChar = (char)Serial.read(); //read the input
      inputString += inChar;        //make a string of the characters coming on serial
    }
    Serial.println(inputString);
    while (Serial.available() > 0)
    { junk = Serial.read() ; }      // clear the serial buffer
    if(inputString == "a"){         //in case of 'a' turn the LED on
      digitalWrite(13, HIGH);
    }else if(inputString == "b"){   //incase of 'b' turn the LED off
      digitalWrite(13, LOW);
    }
    inputString = "";
  }
}
```

f. Senior Design II Program Code

```
       #include <Arduino.h> // Include Arduino library
#include <Servo.h>   // Include Servo library
#include <SoftwareSerial.h> //Include SoftwareSerial Library
#include <Wire.h> //Opens communication with SCL & SDA (For gyro)
#include <SR04.h> //HC-SR04 (Ultrasonic sensor) library. !!Uses delays,inherently!!


Servo LEFT;             // Servos named
Servo RIGHT;            //.

char CONTROLLER;        //BlueTooth
String In_String = "";  //.

void ONWARD(int);       //Movement Functions
void BACKWARD(int);     //
void LEFTWARD(int);     //
void RIGHTWARD(int);    //
void GO_ON(void);       //
void STAY(void);        //.


int main(void)
{

  Serial.begin(9600);  // Data Rate == 9600bps

  int i = 0;  //Tracks gryo readings


  for(;;)
  {
    ////////////////////////////////////////////////////////////////////////IR Sensor + SerialMonitor



    int Left_Higher_IR_Led = 24; int Left_Higher_IR_Receiver = 22;    //Defining IR LED ad Receiver pins
    int Right_Higher_IR_Led = 28; int Right_Higher_IR_Receiver = 26;  //
    int Middle_IR_Led = 32; int Middle_IR_Receiver = 30;              //.


    long Left_Higher_IR_Frequency = 43000;          //Frequency. Higher Hz makes bot near-sighted.
    long Right_Higher_IR_Frequency = 43000;         //
    long Middle_IR_Frequency = 45000;               //.




    tone(Left_Higher_IR_Led, Left_Higher_IR_Frequency, 8);            //Square wave tone...
    delay(1);                                                         // time in ms...
    int Left_Higher_InfraRed = digitalRead(Left_Higher_IR_Receiver);  //Read pin. 0 = item, 1 = no item...
    delay(3);                                                         //
    tone(Right_Higher_IR_Led, Right_Higher_IR_Frequency, 8);          //
    delay(1);                                                         //
    int Right_Higher_InfraRed = digitalRead(Right_Higher_IR_Receiver);//
    delay(3);                                                         //
    tone(Middle_IR_Led, Middle_IR_Frequency, 8);                 //
    delay(1);                                                         //
    int Middle_InfraRed = digitalRead(Middle_IR_Receiver);       //
    delay(3);                                                         //.
```

```
////FOR CALIBRATING IR SENSORS//////
/*if(Left_Higher_InfraRed == 1)  ////TEST IR Top-left
{
  //Serial.println(" ");
}
else if(Left_Higher_InfraRed == 0)
{
  Serial.println("1");
}
else
{
  //Serial.println("Uh oh? 1");
}

if(Right_Higher_InfraRed == 1)  ////TEST IR Top-right
{
  //Serial.println(" ");
}
else if(Right_Higher_InfraRed == 0)
{
  Serial.println("2");
}
else
{
  //Serial.println("Uh oh? 2");
}

if(Middle_InfraRed == 1)  ////TEST IR Middle
{
  //Serial.println(" ");
}
else if(Middle_InfraRed == 0)
{
  Serial.println("3");
}
else
{
  //Serial.println("Uh oh? 3");
}

delay(1000);

Serial.println("");
Serial.println("_");
Serial.println("");*/


////////////////////////////////////////////////////////////////////////////Gyro + SerialMonitor
const int MPU_addr=0x68;
int16_t Gyro_Accel_X, Gyro_Accel_Y, Gyro_Accel_Z ,Tmp,GyroX,GyroY,GyroZ;

int MinimumValue=265;
int MaximumValue=402;

double Gyro_X;  double Gyro_Y;

Wire.begin();                      //Initialize Gyro
Wire.beginTransmission(MPU_addr); //
Wire.write(0x6B);                  //
Wire.write(0);                     //
```

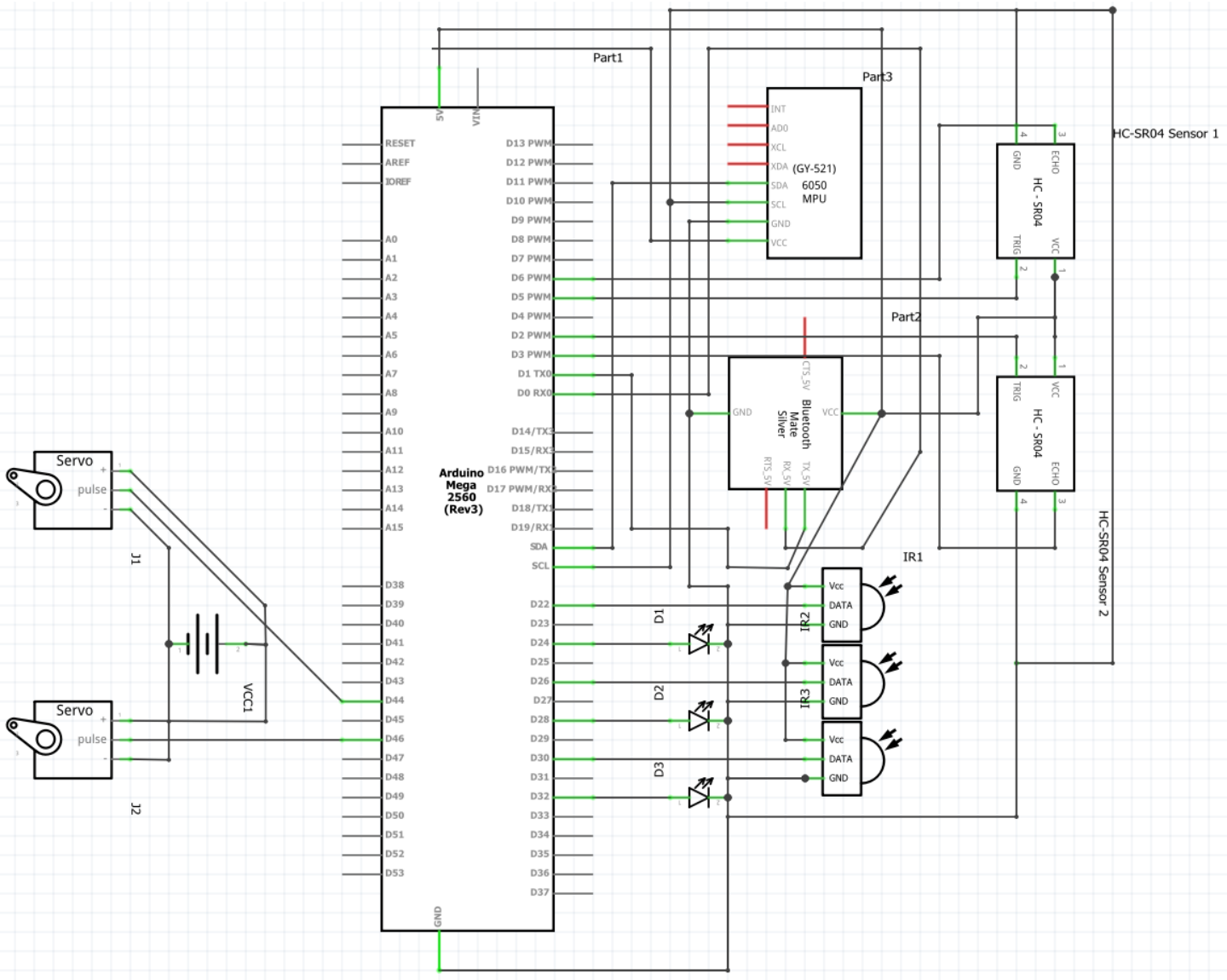
```
    Wire.endTransmission(true);       //.

    Wire.beginTransmission(MPU_addr);             //Set IC2 connection for Arduino and MPU
    Wire.write(0x3B);                             //
    Wire.endTransmission(false);                  //
    Wire.requestFrom(MPU_addr,14,true);           //
    Gyro_Accel_X = Wire.read() << 8|Wire.read();  //
    Gyro_Accel_Y = Wire.read() << 8|Wire.read();  //
    Gyro_Accel_Z = Wire.read() << 8|Wire.read();  //.

    int Gyro_Angle_X =-map(Gyro_Accel_X, MinimumValue, MaximumValue, -90, 90); //MPU data
    int Gyro_Angle_Y = map(Gyro_Accel_Y, MinimumValue, MaximumValue, -90, 90); //
    int Gyro_Angle_Z = map(Gyro_Accel_Z, MinimumValue, MaximumValue, -90, 90); //.

    Gyro_X= RAD_TO_DEG * (atan2(-Gyro_Angle_Y, -Gyro_Angle_Z)+PI)-270; //Give positive values at 90 deg.
    Gyro_Y= RAD_TO_DEG * (atan2(-Gyro_Angle_X, -Gyro_Angle_Z)+PI)-270; //.

    if (Gyro_X < 0) {Gyro_X = -Gyro_X - 180;} //Convert negative values to respective positive 0-90 deg.
    if (Gyro_Y < 0) {Gyro_Y = -Gyro_Y - 180;} //.

    ////FOR TESTING GYRO READINGS SENSORS//////
    //Serial.print("Gyro_Angle_X = "); Serial.println(Gyro_X); ////TEST GYRO X
    //Serial.print("Gyro_AngleY = "); Serial.println(Gyro_Y);  ////TEST GYRO Y
    Serial.println();


    if(Gyro_X < 60 || Gyro_Y < 70)                  //Gyro detection//
    {
      i++;
    }
    else if(Gyro_X > 60 || Gyro_Y > 70)             //Gyro registers clear//
    {
      i = 0;
    }


    ////////////////////////////////////////////////////////////////////Ultrasonic Sensor + Serial Monitor


    long Left_US_TRIG = 2;  long Left_US_ECHO = 3;      //Pins 2 & 3
    long Right_US_TRIG = 5;    long Right_US_ECHO = 6;  //Pins 5 & 6

    SR04 Left_sr04 = SR04(Left_US_ECHO,Left_US_TRIG);     //HC-SR04
    SR04 Right_sr04 = SR04(Right_US_ECHO, Right_US_TRIG); //.

    long Left_UltraSonic = Left_sr04.Distance();    //HC-SR04 distance values
    long Right_UltraSonic = Right_sr04.Distance();  //.

    Serial.print(Left_UltraSonic);  Serial.println("cm(L)");   ////TEST ULTRASONIC LEFT
    Serial.print(Right_UltraSonic);  Serial.println("cm(R)");   ////TEST ULTRASONIC RIGHT
    Serial.println();


////////////////////////////////////////////////////////////////////SAFETY CHECKLIST & CONTROLLER REQUESTS


      if(Left_UltraSonic < 4 || Left_UltraSonic > 7)        //Ledge on left//
      {
        GO_ON();  BACKWARD(1200);

        STAY(); delay(200);
```

```
    GO_ON();  RIGHTWARD(375);
  }
  else if (Right_UltraSonic < 4 || Right_UltraSonic > 7) //Ledge on Right//
  {
    GO_ON();  BACKWARD(1200);

    STAY(); delay(200);

    GO_ON();  LEFTWARD(375);
  }
  else if ( ((Left_UltraSonic <4 || Left_UltraSonic >7))&&((Right_UltraSonic <4 || Right_UltraSonic >7)) )
  {                                                    //Ledge straight on//
    GO_ON();  BACKWARD(1200);

    STAY(); delay(200);
  }



  else if (Left_Higher_InfraRed == 0)                  //Object on Right//
  {
    Serial.print("On right!");
    GO_ON();  BACKWARD(1000);

    STAY(); delay(200);

    GO_ON();  LEFTWARD(375);
  }
  else if (Right_Higher_InfraRed == 0)                 //Object on Left//
  {
    Serial.print("On left!");
    GO_ON();  BACKWARD(1000);

    STAY(); delay(200);

    GO_ON();  RIGHTWARD(375);
  }
  else if (Middle_InfraRed == 0)                       //Object straight on//
  {
    Serial.print("In middle!");

    GO_ON();  BACKWARD(1000);

    STAY(); delay(200);
  }
  else if (Left_Higher_InfraRed == 0 && Right_Higher_InfraRed == 0)
  {                                                    //Object straight on//
    Serial.print("Left/Right says middle!");

    GO_ON();  BACKWARD(1000);

    STAY(); delay(200);
  }



else if( i > 4)                    //High tilt path//
  {
    GO_ON();  BACKWARD(2000);
```

```
        STAY(); delay(200);

        i = 0;
      }


     else if(Serial.available())
      {
        while(Serial.available())
        {
          char In_Direction = (char)Serial.read(); //read the input data
          In_String += In_Direction;        //Incoming serial character form a 'string'
        }

        Serial.println(In_String);

        while (Serial.available() > 0)
        {
          CONTROLLER = Serial.read() ;
        }      // clear the serial buffer

        if(In_String == "G")        //Go forward//
        {
          GO_ON(); ONWARD(500);
        }
        else if(In_String == "L")   //Head Left//
        {
          GO_ON(); LEFTWARD(400);
        }
        else if(In_String == "R")   //Head Right//
        {
          GO_ON(); RIGHTWARD(300);
        }
        else if(In_String == "S")   //Stop movement//
        {
          STAY();
        }

        In_String = "";
      }



    delay(250); ////////////////////////////////////////////////Refresh all measurements
  }


  return 0;
}


void setup()
{
  pinMode(24, OUTPUT);   //Left-Left IR LED          //IR
  pinMode(22, INPUT);   //Left-Left IR Receiver      //
                                                     //
  pinMode(28, OUTPUT);   //Left-Center IR LED        //
  pinMode(26, INPUT);   //Left-Center IR Receiver    //
                                                     //
  pinMode(32, OUTPUT);   //Right-Center IR LED       //
  pinMode(30, INPUT);   //Right-Center IR Receiver   //
```

```
                                                    //
  pinMode(36, OUTPUT);   //Right-Right IR LED        //
  pinMode(34, INPUT);   //Right-Right IR Receiver    //.


  pinMode(2, OUTPUT);   //Left Ultrasonic Trigger    //Utlra Sonic
  pinMode(3, INPUT);   //Left Ultrasonic Receiver    //
                                                    //
  pinMode(5, OUTPUT);   //Right Ultrasonic Trigger   //
  pinMode(6, INPUT);   //Right Ultrasonic Receiver   //.

  //Serial.begin(9600);  // Data Rate == 9600bps
}

////////////////////////////////////////////////////////////////////////////SERVO FUNCTIONS

void ONWARD(int TIME)
{
  LEFT.writeMicroseconds(1625);         // Left -> CWW
  RIGHT.writeMicroseconds(1300);        // Right -> CW
  delay(TIME);
}

void BACKWARD(int TIME)
{
  LEFT.writeMicroseconds(1300);         // Left -> CW
  RIGHT.writeMicroseconds(1700);        // Right -> CW
  delay(TIME);
}

void RIGHTWARD(int TIME)
{
  LEFT.writeMicroseconds(1600);         // Left -> CCW
  RIGHT.writeMicroseconds(1600);        // Right -> CW
  delay(TIME);
}

void LEFTWARD(int TIME)
{
  LEFT.writeMicroseconds(1400);         // Left -> CW
  RIGHT.writeMicroseconds(1300);        // Right -> CWW
  delay(TIME);
}


void GO_ON()
{
  LEFT.attach(46);                   //Start servos (wheels now move)
  RIGHT.attach(44);                  //.
}

void STAY()
{
  LEFT.detach();                     // Stop servos (wheels stay still)
  RIGHT.detach();                    //.
}
```

3. Circuit Design [13]



VII. COGNITIVE UNDERSTANGING OF STUDENT LEARNING OBJECTIVES

1. The ECET program:

a. a)

An appropriate mastery to select and apply the knowledge, techniques, skills and modern tools of the discipline of Electrical Engineering Technology was addressed through the integration of circuit design, C++ language programming, experimentation and diagnostic testing, and varying component incorporation to design and construct a functioning project suitable for Senior Design II presentation.

b. b)

An ability to select and apply current knowledge and adapt to emerging applications of mathematics, science, engineering and technology to Electrical Engineering Technology problems that require the application of principles and applied procedures or methodologies was addressed through the research and application of varying familiar and unfamiliar sensor and microcontroller instruments so as to achieve the project's required objectives.

c. c)

An ability to conduct standard tests and measurements; to conduct, analyze, and interpret experiments and to apply experimental results to improve processes was addressed through the continued study and formulaic alterations of datasets gathered by the project's assortment of utilized sensors. These included sensor data from infrared and ultrasonic sensors, gyroscopes, and Bluetooth devices; further observational data was collected regarding the implementation servo motor utilization within the project.

d. d)

An ability to design systems, components, or processes for broadly-defined Electrical Engineering Technology problems appropriate to program educational objectives was addressed through the research and formulation of a design that incorporated a variety of elements into a complex, functioning project.

e. e)

An ability to function effectively as a member or leader on a technical team was unable to be addressed as I worked alone as the only student in this project. However, I did act as a team member of sorts when discussing project progress and potential project solution concepts with my fellow Senior Design class members each weekly class meeting.

f. f)

An ability to identify, analyze, and solve broadly-defined Electrical Engineering Technology problems was addressed though the formation of solutions and problem-solving strategies to accomplish the project's objective proposals regarding a mixture of elements used in prior classes and items never before practiced with.

g. g)

An ability to apply written, oral, and graphical communication in both technical and nontechnical environments and an ability to identify and use appropriate technical literature was addressed in my presentations and submitted written report.

h. h)

An understanding of the need for and an ability to engage in self-directed continuing professional development was addressed through the research and formulation of a project design that made use of a multitude of varying components that potentially were never utilized in prior class settings. This could further be observed through the necessitated self-dependence required to complete this project as a solo-student application where there was no ability to shrug work or research off to a separate project partner if the subject matter was unfamiliar or complex in nature.

i. i)

An understanding of and a commitment to address professional and ethical responsibilities including a respect for diversity was displayed through the courteous and meaningful communication and idea exchanges with the wide variety of students at Purdue Calumet.

j. j)

A knowledge of the impact of engineering technology solutions in a societal and global context was addressed through the research and project application submission to ASEE.

k. k)

A commitment to quality, timeliness, and continuous improvement was addressed through my work and diligence regarding the project's deadlines and requirements.