Southern Illinois University at Carbondale
College of Engineering
Department of Technology

EET 495
Senior Design Project
Fall 2017- Spring 2018

Control System Design of Remote Control Stage Scenery Robot

Design Team Members

Name	Major	Email
Mohammed Alabdulwahab	Electrical Engineering Tech	m.alabdulwahab@siu.edu
Yasir Alharthi	Electrical Engineering Tech	yasir@siu.edu
Hassan Alkhwaildi	Electrical Engineering Tech	Hassan.alkhwaildi@siu.edu
Austin Bechtel	Electrical Engineering Tech	austinbechtel@siu.edu
Patrick Dudczyk	Electrical Engineering Tech	pdudczyk89@siu.edu
Nick Sjoberg	Engineering Tech	nick.sjoberg@siu.edu

Executive Summary (MA)

A design team from Southern Illinois University EET Program designed and constructed a scenery moving Robot. This Robot must be able to move forward, reverse, and turn around. The end design minimized cost while including systems necessary to complete all the objectives outlined by the client requirements. Many different mechanical and electrical systems are integrated into our design including the frame, joy stick, wireless control, edge detection, and emergency stop.

The robot will be controlled wirelessly through a computer using Xbee modules, then to Arduino Mega. The computer will show all information about the robot such as motor speed, battery life, and emergency stop condition. The robot design will be capable of moving scenery such as couches or other props as required by the Theater activities. to move on stage at different speeds. This design is better than others because it uses off the shelf components and software that is easily obtainable and relatively low cost. The final design was tested and operational by the end of April 2018

The testing demonstrated that the robot could move forward, reverse, and turn around using a joy stick. It also implemented a emergency stop switch and edge detection which were not tested due to lack of time. The user interface shows if the robot about to lose connection, and a bar of battery life remaining. Currently the project budget is at $700.00 which leaves $800 contingency.

Table of Contents

Problem Statement

Included in this document, is all the work that has been done up until now to create a robot that will meet all the requirements needed by the client. The client required enhanced safety features, such as stage edge detection and an E-stop switch. The design team elected to develop the control software and communication on a PC to improve system reliability.

Technical Design

General Operation

The operator uses a USB gamepad to control the robot though a PC. The program on the PC takes this data and sends it to the robot wirelessly using XBEE modules. An onboard Arduino Mega serially receives and interprets the data. The Arduino then sends commands to the motor controllers to move the robot accordingly. There are multiple systems in place to ensure safe operation of the robot.

Safety Systems Overview

There are multiple safety systems implemented on the robot. The first system is a physical Emergency Stop button. This button controls power to the 24v contactor. If the button is pressed, the contactor loses power and cuts the high current power to the motor controller and brakes. This leaves the low power electronics powered so that it can exit ESTOP quickly. Additionally, the Arduino controls a relay that can also interrupt the power to the contactor. This allows the Arduino to trigger an ESTOP based upon multiple conditions.

The Arduino monitors the wireless connection between the PC and itself. If the connection is lost, it will trigger an ESTOP. In addition to this, the operator can use ESTOP buttons on the controller and in software that tell the Arduino to trigger ESTOP. Ultrasonic sensors are also monitored. If any sensor detects an object too close to the robot, the Arduino will trigger an ESTOP. The final safety system is edge of stage detection. The edge detection system has its own section due to its complexity.

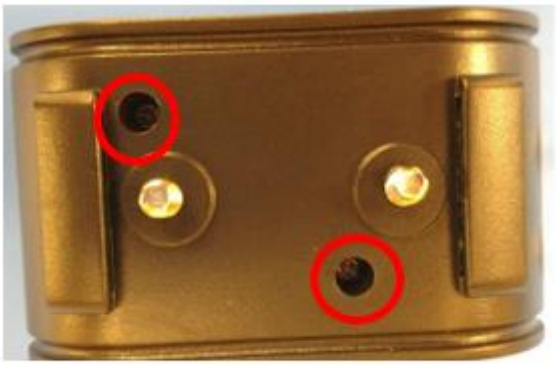
Edge Detection

The purpose of the edge detection system is to maintain safe operation near the edge of the stage. Without any physical boarder on the front of the stage, this became a critical function of the robot to ensure that the robot would not drive over the edge. Since the stage floor color and surface can change for different shows, we found it best to implement a system that is invisible to the audience and lays beneath the stage. To accomplish this function we modified an inexpensive underground dog fence system.

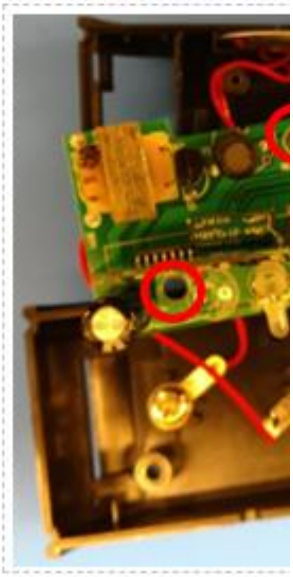
The first step to this process was to remove the transformer that outputs the high voltage shock. Once the transformer is removed, the remaining circuitry is low voltage DC and poses no safety risk. The LED can also be removed at this time so that the pins can be used as the output source from the dog collar. This process is outlined in the steps below.

Step 1: Remove the battery and unscrew the shocking terminals

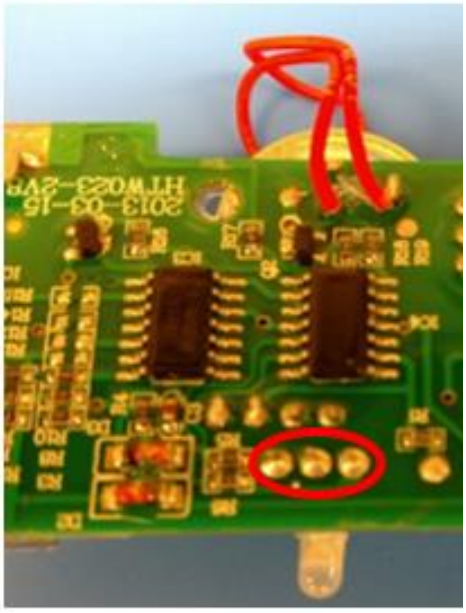
Step 2: Remove the screws that hold the case together



Step 3: Gently pry the case apart and



Step 4: Locate the pins for the tra

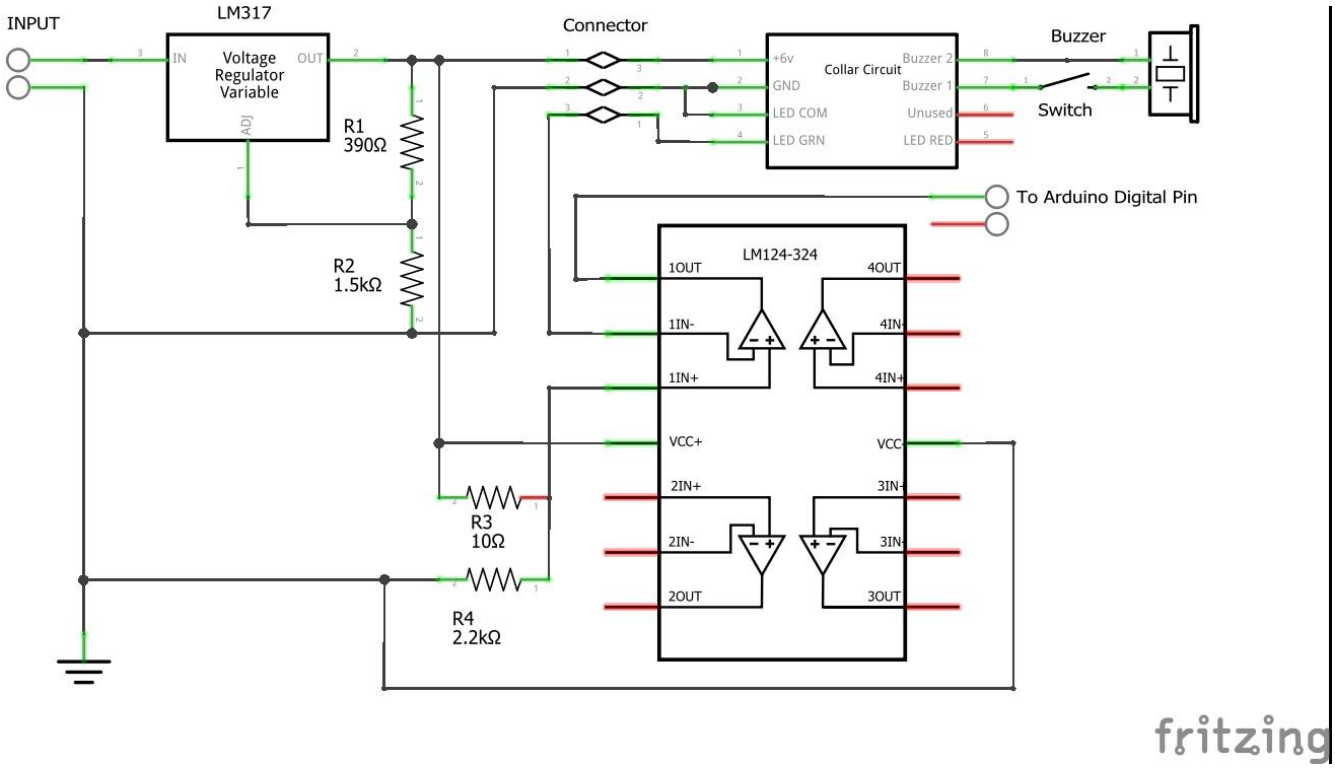


Step 5: De-solder and remov

The contact points for the LED provide an easy to access signal source. However, the signal sent to turn the LED on and off is too low to trigger an Arduino digital pin consistently. A simple circuit, shown in the schematic below, can be used to amplify the signal. The output of the 6v regulator is connected to the battery terminals on the circuit board. This removes the need for the circuit to have its own battery by powering it from the main batteries on the robot. We also added a switch in line with the buzzer. When switched on, the buzzer allows for easy troubleshooting and testing. The

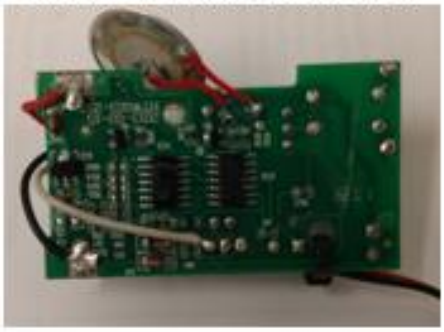
switch allows us to silent the buzzer for shows and rehearsals. The pictures following the schematic show the prototype circuit board and modified collar. A 3 pin JST connector was used to connect the collar circuit board to the amplifier circuit.

Edge Detection Circuit

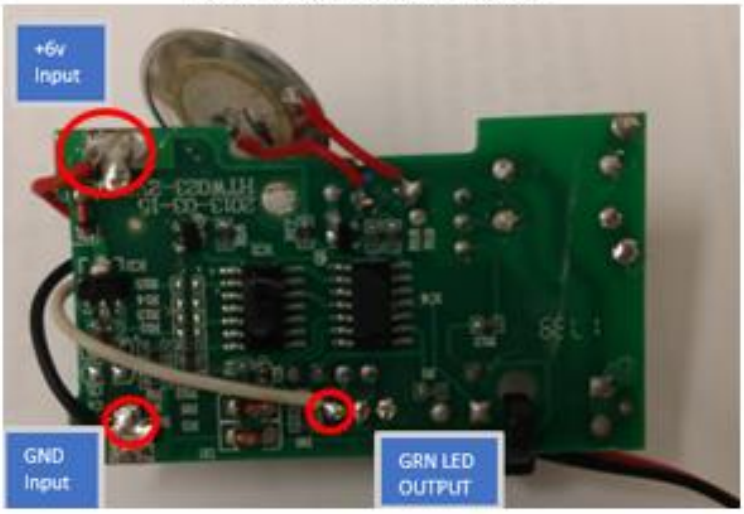


Modified Collar Top View

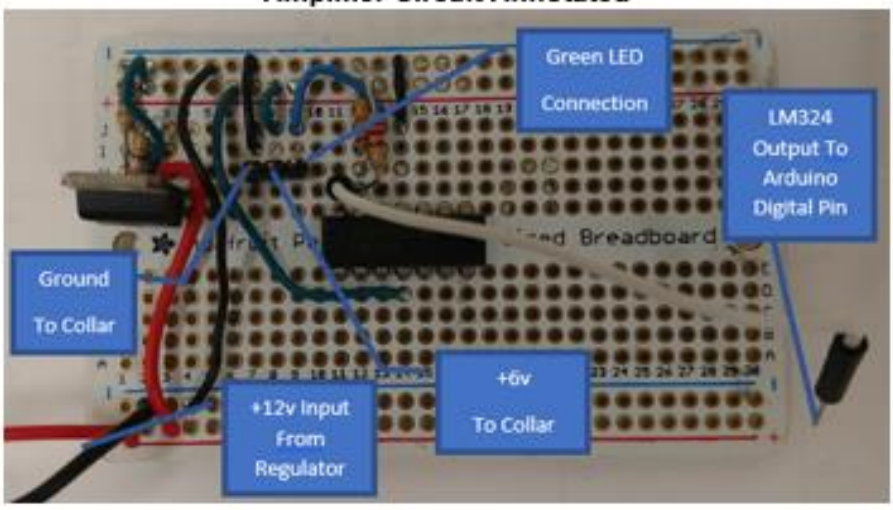
Modified Collar Bottom View



Modified Collar Annotated



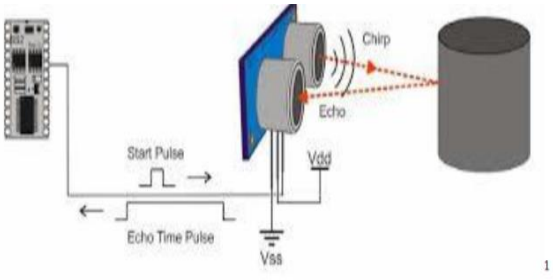
Amplifier Circuit Annotated



Ultrasonic sensor

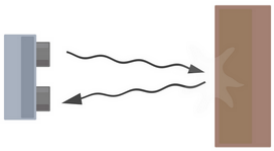
Definition:

The ultrasonic sensor is a device that can detect an object within a limited range (about 3 Meters = 9 Feet). This device transmits a sound wave from the trigger and it does het the object near by this device. Then, it goes back to the echo. If the echo received a signal, we could calculate the distance between the time the trigger went high and the time the sound went back to the echo because the speed of the sound is 344 m/s = 1129 ft/s.



$$distance = \frac{speed\ of\ sound\ \times tim}{2}$$

Benefit from the ultrasonic in our robot:



we can benefit from the ultrasonic in the object detection, where we can calculate the distance and when the object is near by the frame, the Arduino sends an alarm signal to say that the object is near by the frame. If the person whom control the robot keep moving the robot towards the object and didn't pay attention to the alarm signal the robot will activate the E-stop after a specific distance.

The ultrasonic sensor will be attached to each corner in the frame and there will be a several extended wires in which it purpose is to extend the sensor if the robot has an object on it that is more than the length of the robot itself. We have developed our code for the ultrasonic and tested several times to check it is working.

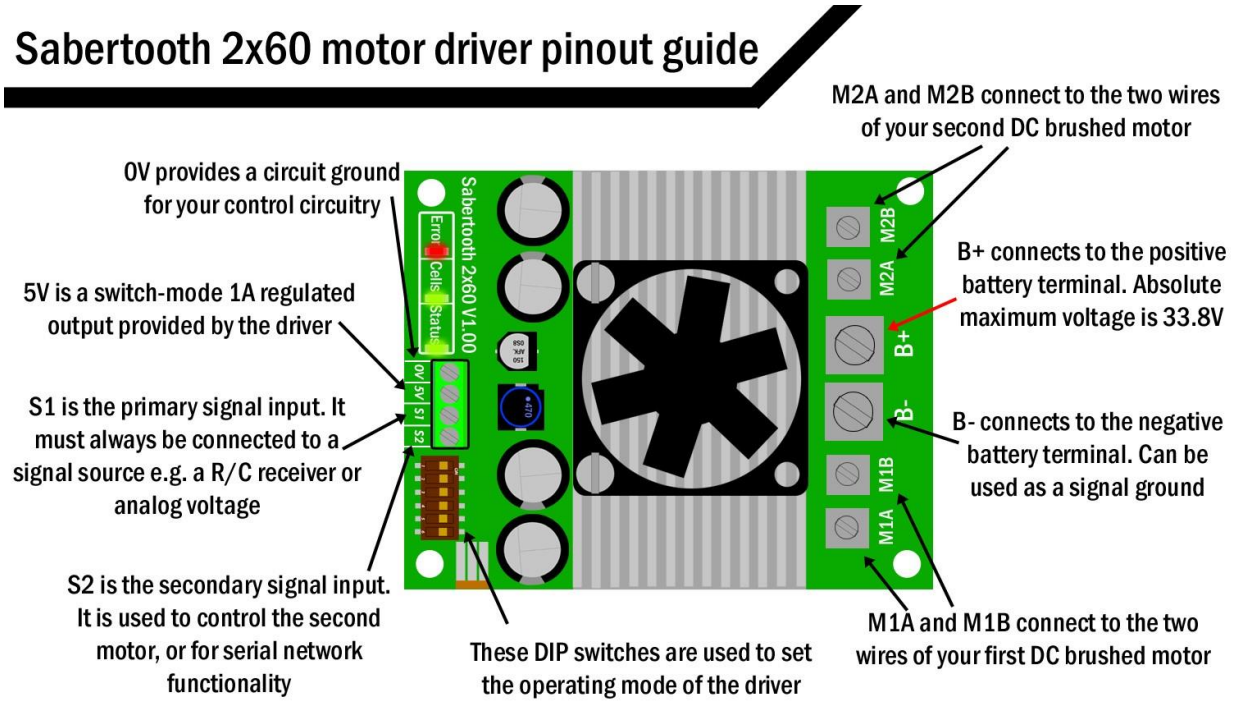
Limitation of Ultrasonic sensor:

There are two main limitation in the ultrasonic sensor

- Distance: the maximum distance that the ultrasonic sensor can detect is 3 Meters
- Some object might not be detected: the ultrasonic sensor might not detect all of the object that it is near by the devise because it is accessible for the sound wave to go through such as cloth or carpeting.

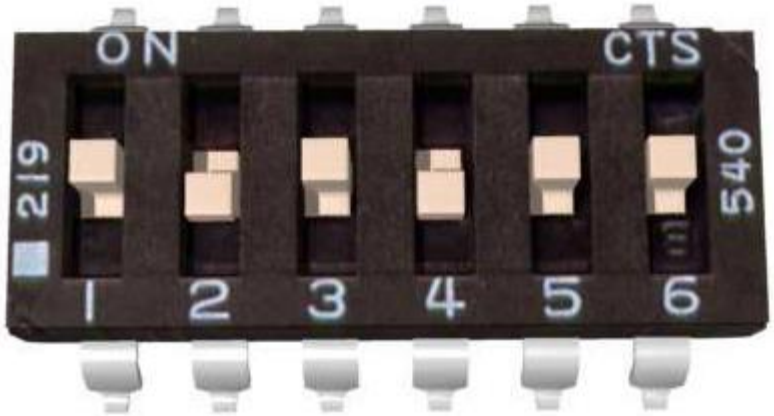
Motor and Brake Control

The motor controller used to control the drive wheels is a Dimension Engineering dual 60 motor driver. This controller is capable of being controlled with Pulse Width Modulation (PWM) analog voltage, and over a serial network. This controller is capable of continuously delivering 60 amps to each motor at up to 33.8 volts. The figure below shows the features of the motor drive.



Do not connect B+ and B- backwards! Make sure you configure the DIP switches properly before connecting power!

The Sabertooth motor controllers have a wide variety of settings that can be configured using the DIP switches located next to the signal inputs. Dimension Engineering has an interactive DIP switch configuration tool. This tool is linked in the references section. The switch positions that we used are shown in the following image. The settings are: Non-Lithium, TTL level RS-232, Simplified Mode, Just One, 9600 Baud Rate. When using simplified serial mode, the Arduino TX pin is connected to S1 on the motor controller. The Arduino RX pin is connected to S2. For better communication, it is recommended to connect the Arduino ground to 0v on the motor controller. **DO NOT CONNECT THE SABERTOOTH 5V TO THE ARDUINO**



The Jazzy Select motors used on this robot have built in brakes. In order to disengage them, 24v must be supplied to the brake wires. As shown in the overall schematic, the brakes and motor controller are powered through the contactor. When the contactor is powered, the high current 24v power is delivered to the motor controller and brakes. This means that any time the motor controller is powered, the brakes should be disengaged. The brakes must be disengaged when the motor controller is on to prevent damage to the motor controller. If the brakes were on when the motor controller tried to turn the motors on, the motors would stall and draw extremely high current from the motor controller, likely causing irreversible damage.

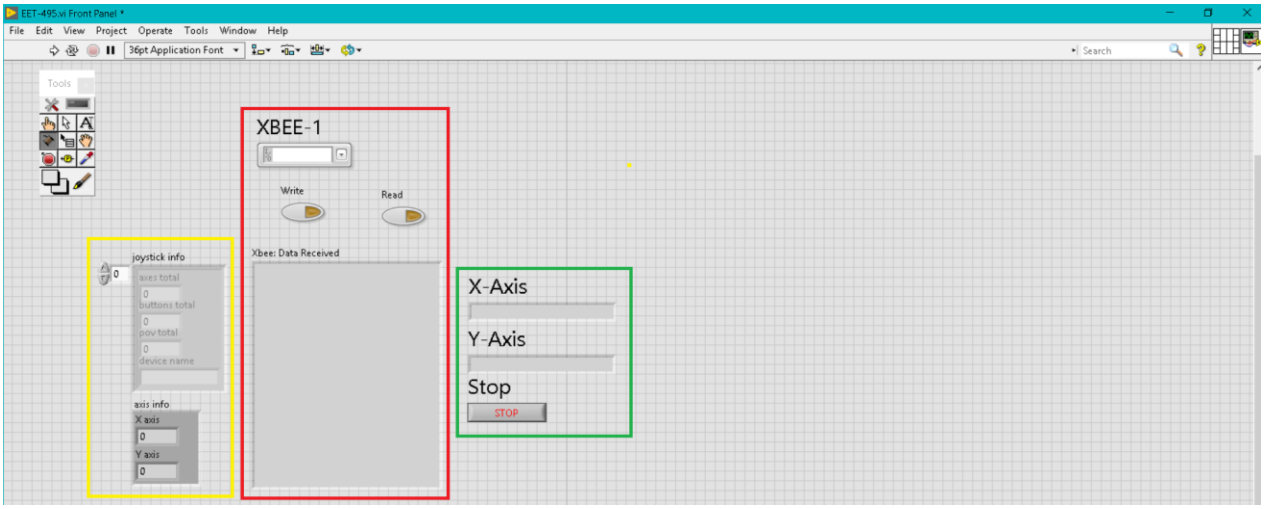
PC Control Program

A control program located and running on any PC provides the user interface and control for the Robot. The user interface gives an operation status information on the robot and links the joystick controller to the wireless link. The LabVIEW program system was selected for this application due to it extensive control and interfacing capabilities.

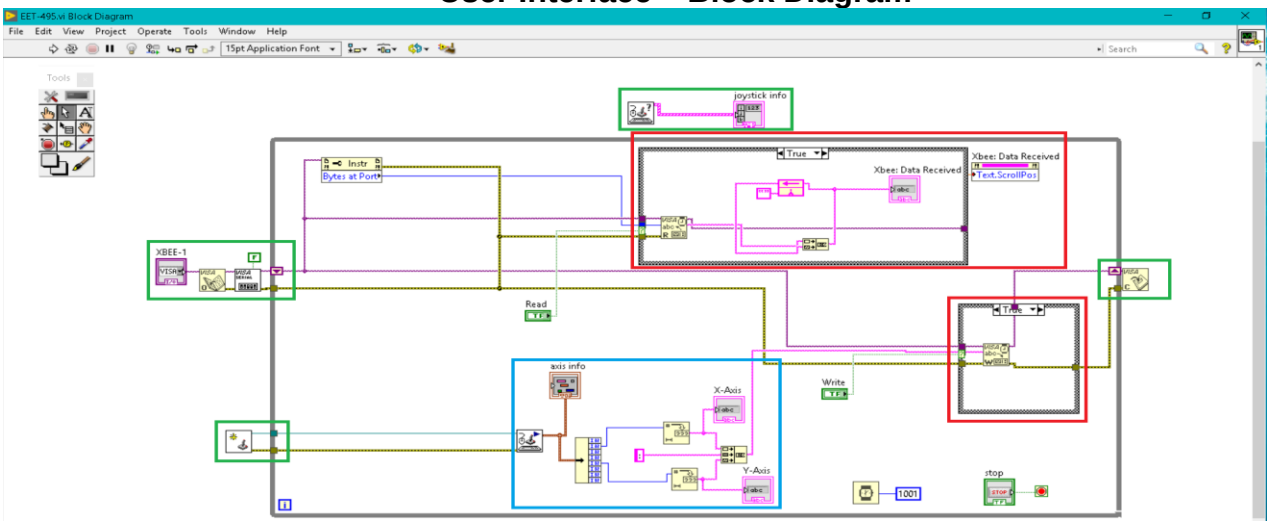
LabVIEW is a graphical programming approach with utilizes visual modules to easily create complex programs. When LabVIEW is started, it opens a blank template known as a VI, or virtual instrument. The VI has two primary windows that work together – the block diagram and the front panel. The block diagram deals with the major programming elements, such as arithmetic, Boolean, instrument I/O, and much more. The front panel is the user interface for the VI, which contains palettes such as controls, terminals and indicators. Additionally, the Arduino and VISA toolkits (plug-ins) were used to perform supplementary tasks not included in the base software.

User Interface

User Interface – Front Panel



User Interface – Block Diagram

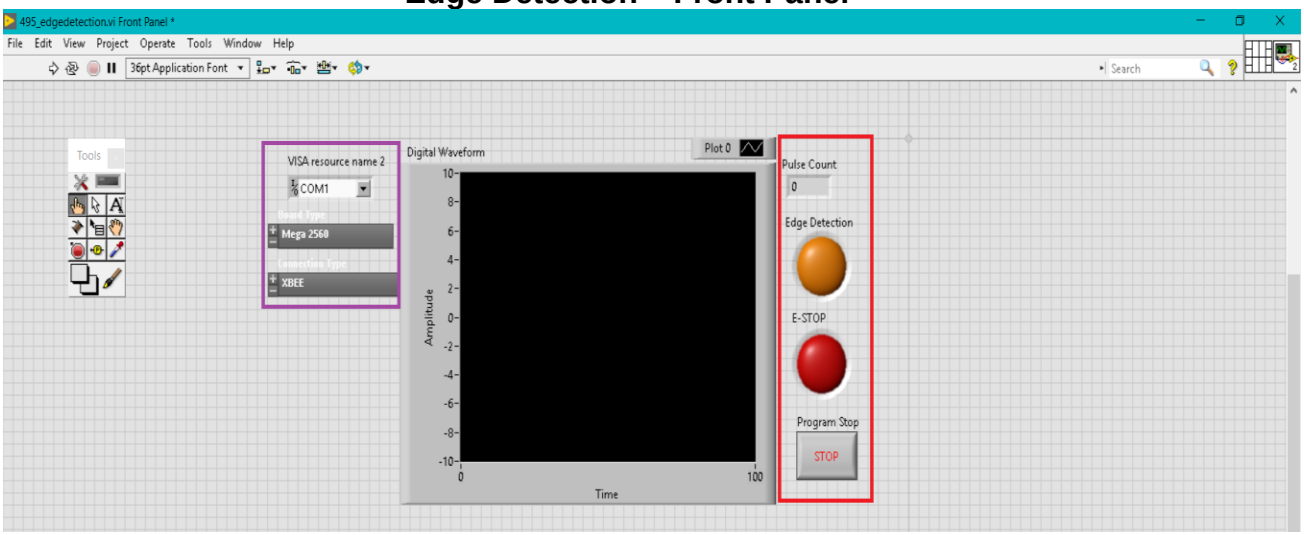


The objectives of the User Interface was to acquire input data from the Logitech L310 gamepad and transmit the serial positioning information wirelessly from one XBEE transceiver to another. The front panel contains all indicators. The user interface's main controls are denoted by the different colored rectangles. The red rectangle contains the serial port (USB) configuration, read & write ON/OFF controls, and an indicator to read incoming (Rx) data in the form of a string data type. The yellow box presents informative indexes, which shows real-time descriptions about the positions of both X-Y axes and a general characterization of the joystick used. The green rectangle displays the synchronous position data acquired from the gamepad. The block diagram contains all major functions needed for the program to run efficiently. The four green boxes show all initialization functions, including the XBEE serial communication and the joystick setup. The teal rectangle deals with acquiring various axes motion information, provided by the

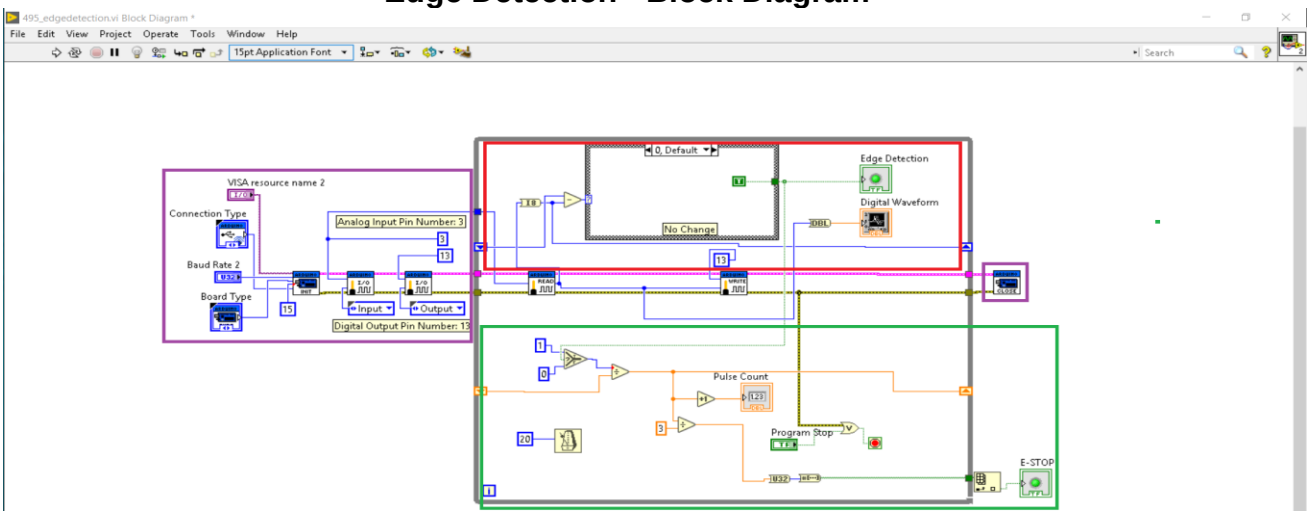
user's input into the gamepad. The final, red rectangle performs the serial read/write actions needed for constant two-way communication between each of the XBEE's.

Edge Detection

Edge Detection – Front Panel



Edge Detection - Block Diagram



The purpose of the Edge Detection VI was to monitor a number of consecutive pulses (1's) read from the hardware. It works in unison with the modified dog collar circuit. The front panel is composed of all necessary indication devices. The purple box displays the selectable initialization parameters for the Arduino, including board type and mode of transmission. A digital waveform is included to show a real-time visual representation of the input wave. The red rectangle presents the number of acquired pulses, the 'Edge Detection' yellow warning indicator. Once a certain number of pulses are collected, the red 'E-STOP' light illuminates and sends a signal to the E-STOP to stop the robot before any damage can be done. The block diagram is made up of all important controls, tasks or functions. The purple selection contains all necessary initialization parameters for the Arduino microcontroller, such as the USB Port

(Resource Name), Connection Type, Baud Rate, Board Type, Stop Bits, Flow Control and I/O Pins. Much like the User Interface VI before, the Edge Detection VI acquires data types of the string variety. The top, red rectangle is concerned with taking the input signal, comparing its Boolean value (0/1) to a true/false case structure. The final, green rectangle, located on the bottom of the block diagram, will take the signal coming out of the case structure, and run in through a few arithmetic and comparison functions. If the signal is HIGH (1), it will be added to 'Pulses' count and the program will continue. If the signal is FALSE (0), the signal will be not collected. The selector function, from mathematic function palette, was employed to ensure that the collected signals correctly travel to their proper destination.

Arduino Programming

Sabertooth Code

```
#include <SabertoothSimplified.h>
SabertoothSimplified ST;
void setup() {
  Serial1.begin(9600);
  SabertoothTXPinSerial.begin(9600); // This is the baud rate you chose with the DIP switches.
  ST.drive(0); // The Sabertooth won't act on mixed mode until
  ST.turn(0);  // it has received power levels for BOTH throttle and turning, since it
}     // mixes the two together to get diff-drive power levels for both motors.So, we set both to zero initially.
void loop() {
#define INPUT_SIZE 30
// Get next command from Serial (add 1 for final 0)
char input[INPUT_SIZE + 1];
byte size = Serial1.readBytes(input, INPUT_SIZE);
// Add the final 0 to end the C string
input[size] = 0;
// Read each command pair
char* command = strtok(input, "&");
while (command != 0){
    // Split the command in two values
    char* separator = strchr(command, ':');
    if (separator != 0){
        // Actually split the string in 2: replace ':' with 0
        *separator = 0;
        int xval = atoi(command);
        ++separator;
        int yval = atoi(separator);
 xval = map(xval, -32768, 32768, -50, 50);     // scale it to use it with the servo (value between 0 and 180)
 yval = map(yval, -32768, 32768, -50, 50);
 ST.motor(2,xval);
 ST.motor(1,yval);
 delay(10);}
    // Find the next command in input string
    command = strtok(0, "&");
}
}
```

The code shown above establishes the serial communication from the User Interface on LabVIEW to the Sabertooth motor controller. This code implements the Sabertooth library and sets the baud rate to 9600. The movement commands being sent from the LabVIEW VI are both x and y coordinates, this code allows the Arduino to decipher the data and accordingly maps the numbers to a more reasonable integer value, that the Sabertooth motor controller can handle and respond to. The maximum reverse and forward speeds can be adjusted by changing the values that are -50 and 50 respectively in the map functions.

Ultrasonic Code

```
String a; //estblashing string for Xbee comms
  const int trigPin = 8;  //pin for ultrasonic
  const int echoPin = 9;   //pin for ultrasonic
long duration;  //vairables for ultrasonic
int distance; //vairables for ultrasonic
void setup()
{
Serial.begin(9600); // opens serial port, sets data rate to 9600 bps, starts serial communication
Serial1.begin(9600);
  pinMode (trigPin, OUTPUT);    //set first pin as output
  pinMode (echoPin, INPUT);   //set second pin as input
}
void loop()
{
while(Serial1.available())
  {
  a= Serial1.readString();// read the incoming data as string
Serial.println(a);
  }
 digitalWrite (trigPin, LOW); //clear pin
 delayMicroseconds (2);
 digitalWrite (trigPin, HIGH); //set high
 delayMicroseconds (10);    //delay of 10microseconds
 digitalWrite (trigPin, LOW);
 duration = pulseIn (echoPin, HIGH); //finds that duration it took for sound wave to travel unit of microseconds
 distance = duration*.034/2;  //calculate distance (uses sepped of sounds which is 340 m/s and speed equalling distance divided by time
 //allows for distance to be find and then divide by 2 because we need to know the distance from sensor to object not sensor to object and back.
 Serial.print ("Distance: "); //serially prints distance that was previously calcualted
  Serial.println (distance);
}
```

This section of code establishes the communication and set up of the ultrasonic sensors. First, the variables of duration and distance were created, which allows us to utilize these variables in a useful way later in the code. The time from the sensor being set to HIGH from LOW is set equal to duration. The distance is then calculated by using the time that was just found, the speed of sound divided by two. The speed of sound is divided by two, as we need to know the distance from the sensor to the object, not the distance from the sensor to the object and back. Therefore, once the code processes, the Arduino will know the relative distance the robot is from any surrounding objects.

Battery Voltage Indicator Code

```
int analogInput = 1;
float Vout = 0.00;
float Vin = 0.00;
float R1 = 3300.00; // resistance of R1
float R2 = 2200.00; // resistance of R2
int val = 0;
void setup(){
   pinMode(analogInput, INPUT); //assigning the input port
   Serial.begin(9600); //BaudRate
}
void loop(){

   val = analogRead(analogInput);//reads the analog input
   Vout = (val * 5.00) / 1024.00; // formula for calculating voltage out i.e. V+, here 5.00
   Vin = Vout / (R2/(R1+R2)); // formula for calculating voltage in i.e. GND
   if (Vin<0.09)//condition
   {
     Vin=0.00;//statement to quash undesired reading !
  }
Serial.print("\t Voltage of the given source = ");
Serial.print(Vin);
delay(1000); //for maintaining the speed of the output in serial moniter
}
```

The battery voltage indicator, shown above, is another segment of the C++ code. Using simple Ohms Law formulas and calculations, this section of code serial prints the estimated source voltage. This allows the user to have a better idea of run time and battery life remaining of the robot. The code can be adjusted for any battery's rated voltage value. In line 13, Vout = (val * 5.00) / 1024.0, the 5.00 represents a 5V battery source. By changing this variable, the program can adjust and work with any power supply or battery, the only requirement is that the numbers must match.

Conclusion

The Electrical Engineering Technology Senior Design Team was brought together to learn how to work effectively as part of a team, or by themselves. Through collaboration with the Theater Department, the students learned that people, despite their background or experience, can come together to achieve a technological design. The objective was to enhance the Theater Department's Robot, by adding motion features and edge detection. The robot moves stationary scenery pieces for various productions. The design team was split into groups of two, one devoted to hardware & another focusing on software. The hardware team concentrated on the edge detection, ultrasonic sensors, safety precautions, and motor & brake control. The software team directed their attention to the motor control, edge detection, and the user interface. The prototype works as follows, the User Interface reads the controllers inputs serially. The data was in the form of a String datatype. The information is again sent serially to the XBEEs, which transmits wirelessly to one another. Once the data is received on the Robot, it will begin executing motion controls, such a forward, back, break and rotate. Overall, the experiences each team acquired will directly assist students in the workplace. Through collusions with different people, each member has a substantial about of technical knowledge and experience.

Reference List

LabVIEW Virtual Instrument Engineering Workbench -
http://www.ni.com/tutorial/13413/en/
LabVIEW VISA Toolkit -
http://search.ni.com/nisearch/app/main/p/bot/no/ap/tech/lang/en/pg/1/sn/catnav:du,n8:3.1637,ssnav:sup/
LabVIEW Arduino Toolkit -
http://www.ni.com/gate/gb/GB_EVALTLKTLVARDIO/US
Arduino IDE -
https://www.arduino.cc/en/Main/Software
XCTU XBEE Configuration Platform -
https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu
Dimension Engineering Sabertooth Serial Arduino Library
https://www.dimensionengineering.com/info/arduino
Dimension Engineering Sabertooth DIP Switch Configuration
https://www.dimensionengineering.com/datasheets/SabertoothDIPWizard/start.htm

Appendix

The final schematic is shown below. A clear image and the original file are both supplied in the EET Research folder on the OneDrive. The digital pin connections may not match the pin numbers in the example code.