University of Portland

Pilot Scholars

Engineering Undergraduate Publications, Presentations and Projects

Shiley School of Engineering

Spring 2020

Accessible Vehicle for Kids

Andrew Shyne

University of Portland
School of Engineering
5000 N Willamette Blvd.
Portland, OR 97203-5798

Phone: (503) 943-7292
Fax: (503) 943-7316

Senior Honors Project

Accessible Vehicle for Kids

Honors Team Member:
Andrew Shyne

Team Members:
Rebekah Bourgeois
Jessica Cherry
Kori Hansen

Faculty Advisor:
Dr. Joseph Hoffbeck

Industry Advisor:
David Laning

Table of Contents

Tables of Figures

Tables of Tables

Introduction

The purpose of this document is to outline the process of developing a Go Baby Go vehicle for use by children with disabilities who experience limited or delayed mobility. The purpose of the project is to adapt the *Wild Thing*, a 12V commercially available ride-on toy car, so that young children with limited mobility can have similar opportunities to independently and actively explore their world for participation, play, learning, and engagement like their same-aged peers. The motivation for undertaking this project is to create a safe and universally designed form of active mobility for children who do not have access to a power wheelchair due to the lack of availability of appropriate products and funding sources for children with disabilities. The original design of the *Wild Thing* has two joysticks which can be moved by pushing the joysticks either both forward to go forward, both backwards to go backwards and one forward and one backwards in order to spin left or right. Because of this original design, children with limited mobility could have a difficult time navigating the *Wild Thing* platform. In order to meet this need, the vehicle will be redesigned to support varying options to control the vehicle. These include a single joystick, head array buttons, or hand-pushed buttons. Dependent on the child's needs, these options can be chosen from, and implemented as the user sees fit. An app will also be utilized that allows control of the vehicle through the use of directional buttons and an emergency stop button. The app will also have the ability to log the usage data of the vehicle. This document contains the design aspects and process of this project.

Design Outcomes

High-Level Architecture

Below is a high-level block diagram of the modifications to be implemented on the Wild Thing. Figure 1 has all the major components included and below the figure is an explanation for each. The major components of this design include the battery, an emergency shut-off, the Wild Thing's control board, a microcontroller, a phone, and a user interface. This snippet of seemingly technical information has been included to offer a better illustration of what the project is going to accomplish and how it will do so.

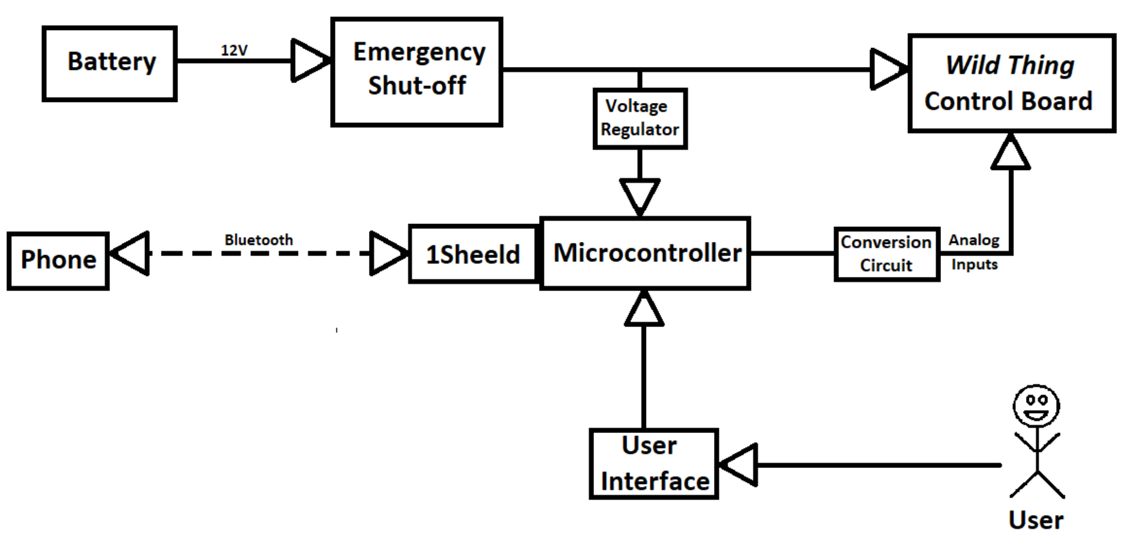

Figure 1 - Block Diagram

Battery – The battery provides 12V and is rechargeable. It came with the Wild Thing.

Emergency Shut-Off – This will be a switch located on the back of the Wild Thing that will cut the power from the battery causing the vehicle to come to a complete stop.

Microcontroller – The microcontroller will get power from the battery after the voltage is stepped down with a voltage regulator. Once it's powered it will receive information from the user interface and then send the correct signals to the control board of the Wild Thing. It will also receive info via Bluetooth from the parent's phone. The microcontroller will consist of an Arduino and a 1Sheeld+. The Arduino is a physical programmable board that also comes with a software component for easier understanding and use. The 1Sheeld+ will allow for Bluetooth connection with a phone. It comes with an application that will allow the parent to have control of the vehicle and log the usage data of the vehicle.

Conversion Circuit – The conversion circuit is four different RC circuits (Figure 5 below) that will convert the outputs of the microcontroller, which are pulse width modulation (PWM), to analog which the Wild Things control board takes. There were problems with driving the load and there not being a high enough current so an opamp was added to help increase the current delivered.

User Interface – The user interface will be tailored to what the user needs. It will be either a single joystick, two buttons, or a head array of buttons. It will receive input from the user and send it to the microcontroller which will process the information and send the necessary commands to the *Wild Thing* control board to make the vehicle move according to the user input.

Phone – The phone will have an application that will allow the parent to control the movement of the vehicle as well as an emergency stop button. The emergency stop will stop the vehicle by sending the rest voltages to the microcontroller and will keep the user from being able to use the interface until it is reset.

User Interface

There are two ways the Accessible Vehicle can be controlled; they are the Joystick and the 1Sheeld+ app. The Joystick interface is replacing the current two joysticks that are located on either side of the car with one joystick located on the right side of the *Wild Thing*. This new joystick would be a four-way joystick (up, down, right, and left). This will allow for the user to be able to move the vehicle forward, backward, spin right and spin left. The second interface is the 1Sheeld+ smartphone application. The 1Sheeld+ is used by a handheld phone app that works with the Arduino Uno to be able to implement motion controls and an emergency stop through the app. This would allow for the parent to be able to control the vehicle and stop the vehicle if they need to.

The 1Sheeld+ plugs into the top of the Arduino and the Arduino board is setup the same. On the 1Sheeld+ app, available on IOS and Android, the parent will designate the left arrow as pin 4, the up arrow as pin 7, the right arrow as pin 2, and the down arrow as pin 12. As the buttons are pressed on the smartphone, signals will be sent to the proper Arduino inputs which would send the proper outputs to the *Wild Thing* control board, and the car will be in the control of the parent. The user interface through the 1Sheeld+ app is shown below.

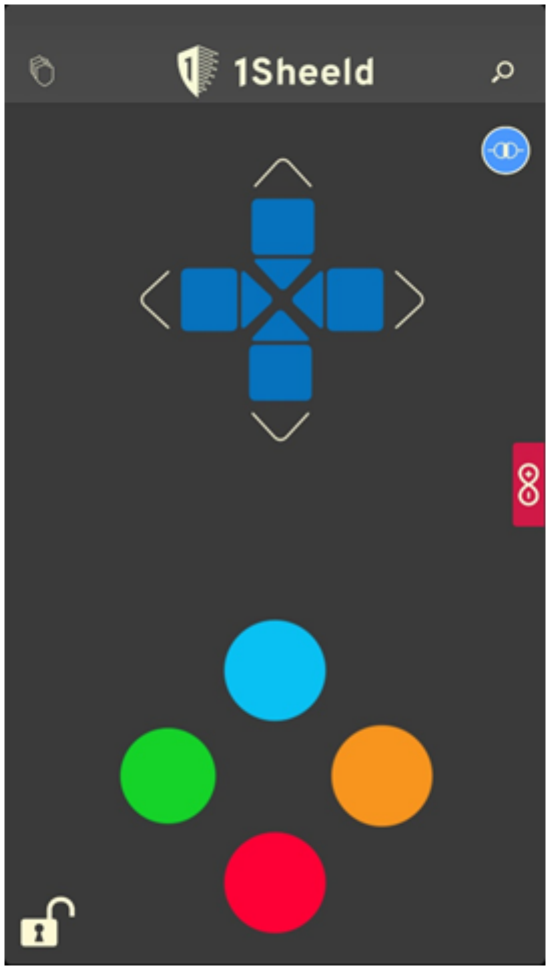

FIGURE 2 - 1SHEELD+ USER INTERFACE

The 1Sheeld+ will also have data logging capabilities. The “Data Logger” shield on the 1Sheeld+ smartphone application will need to be activated by the user and it will work seamlessly. In order to report the data to Bethany Sloan (Go Baby Go ambassador and client), the parent will go the Data Logger shield in the application, click on the “settings gear” on the right edge of the screen, click on the “share box” on the top right of the screen and share it according to further instruction. Note that the Data Logger functionality has only been tested with IOS devices.

There has been a very strong push for the implementation of the Data Logger functionality on the vehicle because it will help Go Baby Go improve their future vehicles. They will be able to see what is used and what is not, how often the car is used by the child and hopefully be able to use this data to improve the experiences of future children.

Component Structure

Below are the different types of component structure.

Software Component

The Accessible Vehicle for Kids project is dependent on an Arduino program, written by the team, that takes inputs from the joystick or other user interfaces and 1Sheeld+ application and outputs the corresponding voltages to mimic the original W*ild Thing* controllers. The vehicle moves in four distinct motions, it goes forwards, backwards and spins left and right. In order to make the vehicle spin left, voltages are sent to the *Wild Thing* controller such that the right wheel goes forwards and the left wheel goes backwards and vice versa for a right command. As safety is the number one priority in this project, it is important that the parent has ultimate control of the vehicle. The Arduino code is written such that when the 1Sheeld+ sends a command, the Arduino will take it at the highest priority. When the emergency stop button is pressed on the 1Sheeld+ app, the *Wild Thing* will stay at rest and not be able to move until the *Wild Thing* is turned off and on again. If connection between the 1Sheeld+ and the phone is ever lost, the vehicle will remain in a rest state and will not give control to the child.

The code will loop, and while there is no input detected, the outputs will report the "at rest" state of the vehicle. When an input is detected from either the joystick or the 1Sheeld+ application, the Arduino will output the proper voltages according to the input selected.

Analog Circuitry

The project has three circuits necessary for safe and proper implementation. The first is a voltage regulator component that will be used as a step-down so the Arduino microcontroller can be safely powered, the second is a set of RC circuits that average out the pulse width modulation (PWM) signals from the Arduino, and the third is a switch that will cut power to the system allowing the parent to manual stop the vehicle in case of an emergency. The PWM signals are created by sending a square wave what alternates from 0V to 5V. The analog voltage is reported in the average of the PWM signal. So in order for the average voltage to be greater than 2.5V, more time is spend on the 5V section of the square wave and vice versa.

Voltage Regulator

The voltage regulator circuit will convert the car battery voltage from 12V to 9V, this will allow us to draw power from the battery to power the Arduino and 1Sheeld+. This is ideal because it allows us to only rely on one, rechargeable power source for the entire system. (See Figure 3)

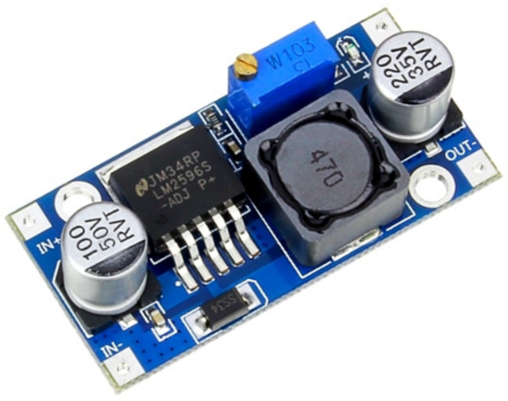

FIGURE 3 - VOLTAGE REGULATOR

In order to properly configure the Voltage Regulator (LM 2596) we attach the input ports to a power supply of +12V and read the output ports with a DMM until it reads +9 V while spinning the brass flathead screw on the LM 2596, after which, the voltage supply will be ready for assembly.

RC Circuits

The circuits that will be used are RC circuits that will average out the PWM signals from the Arduino outputs. When designing, it was important that the speed of the *Wild Thing* ramps up smoothly to provide a more enjoyable experience for the user. To meet this goal, use larger capacitors on the speed control inputs. The larger the capacitor the longer it will take to charge and the slower the voltage will be delivered to the *Wild Thing* control board. This means the car will be going at full speed about a half second after it starts moving. (see Figure 4)

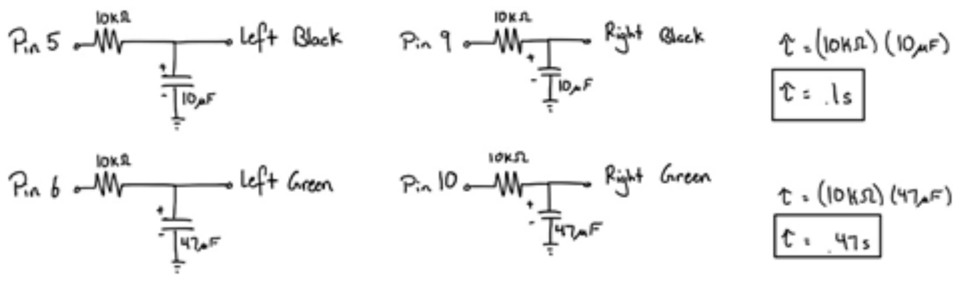

FIGURE 4 - RC CIRCUITS

Since the *Wild Thing* control board takes true analog signals, it is necessary to turn the PWM signals into steady analog signals, an example of this is shown in Figure 5. An example of the PWM to true analog signal is shown below. In Figure 5 there is a PWM signal (+/-5V square wave signal) that averages out to 3V. The state shown is when the joystick is pressed forwards.

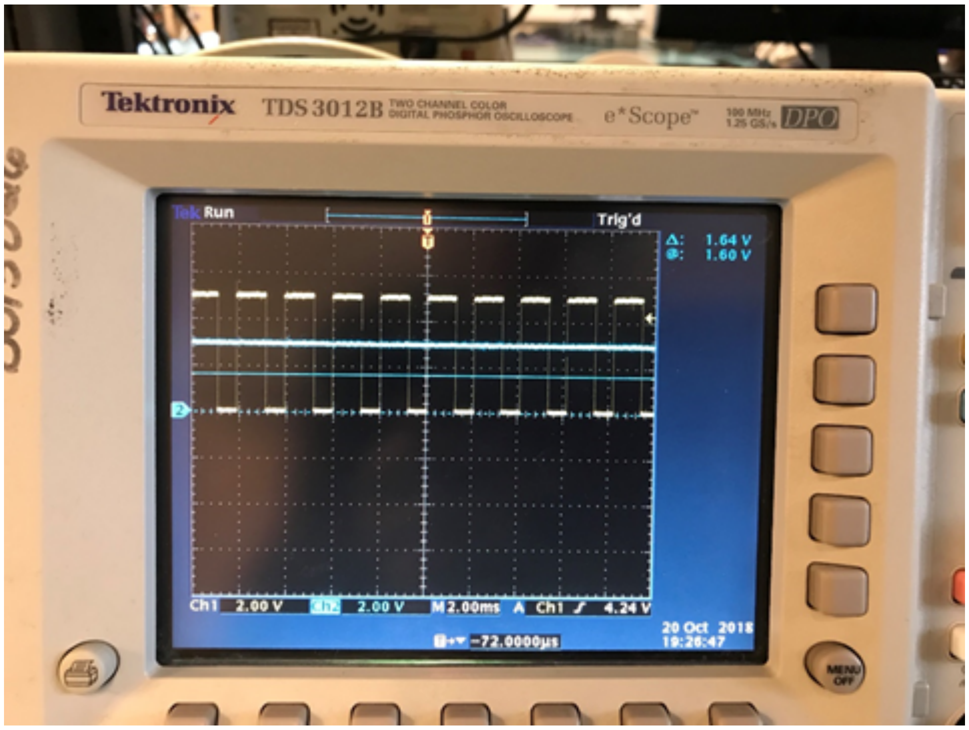

FIGURE 5 - PWM TO TRUE ANALOG

Emergency Shut-Off Switch

There will be a switch located on the back of the wild thing that, when turned off, will cut power to the *Wild Thing* Control Board and the Arduino. This is a key safety feature that allows the parent to stop the vehicle in the case of an emergency.

LM 324 Op Amp

There will be an LM 324 Op Amp following the RC circuits so that the signals from the Arduino/1Sheeld+ are able to drive the load on the *Wild Thing* microcontroller. LM 324 is a single supply +5V Op Amp. Without this component there is not enough current to drive the signal so it is not recognized by the *Wild Thing* control board.

A system schematic that shows all hardware of the design is shown below to give the reader a better picture of how the design is implemented.

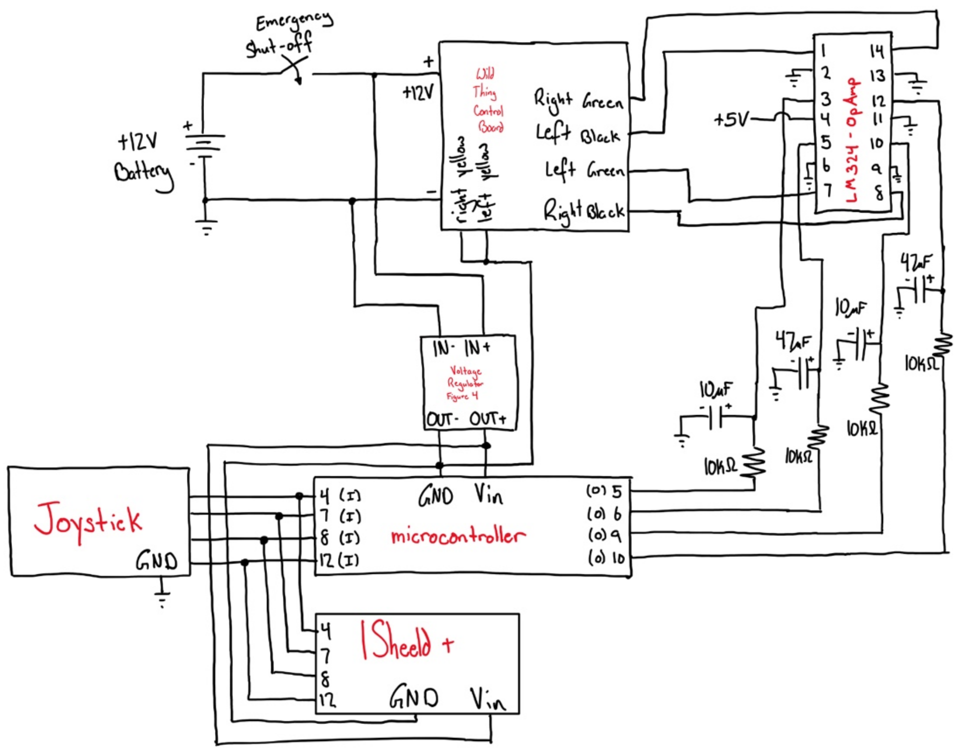

FIGURE 6 - SYSTEM SCHEMATIC

Digital Systems

Original Joystick Signals

It is important to understand the original *Wild Thing* controllers in order to properly mimic them with the Arduino. Each of the original controllers have a five-pin cable that hold the signals below, according to user input.

TABLE 1 - ORIGINAL WILD THING CONTROLLER LEFT

Controller State	Yellow	Black	Green	White	Red
At Rest	0V	1.5V	1.64V	3.3V	3.3V
Forward	0V	2.68V	3V	3.3V	3.3V
Backward	0V	2.68V	0V	3.3V	3.3V

TABLE 2 - ORIGINAL WILD THING CONTROLLER RIGHT

Controller State	Yellow	Black	Green	White	Red
At Rest	0V	1.5V	1.64V	3.3V	3.3V
Forward	0V	2.68V	0V	3.3V	3.3V
Backward	0V	2.68V	3V	3.3V	3.3V

Yellow is constant Ground, White and Red are both constant 3.3V power and Black and Green vary depending on user input. If the controllers are touched at all, the Black signal goes to 2.68V. The Green signal varies from its rest state to its forward/backward state depending on how far the original controller is pressed. In our design we are slowly ramping up/down the voltage using an RC circuit with a time constant of about half a second.

The Arduino is the main component of our design. Its programming will cause the correct voltages to be delivered to the control board making the vehicle move as expected.

The Arduino will read the digital inputs from the joystick. It will check to see which input is triggered. Depending on which input is on, the correct voltages will be sent from the outputs. The outputs are PWM, but they will be converted to analog signals by an RC circuit (Figure 4).

The following is a pseudo-code section that will hopefully give the reader a better understanding of what is happening inside the Arduino code and programable board.

If up is HIGH, both black outputs will go to 2.86V while the left green will go to 0 and right green will go to 3V.

If right is HIGH, both blacks will go to 2.86V while both greens will go to 0.

If down is HIGH, black go to 2.86V while left green goes to 3V and right green goes to 0.

If left is HIGH, both black will go to 2.86V while both greens go to 3V.

If none of the above (up, right, down, left) are on, the rest voltages will be sent. Both blacks will be at 1.5V and both greens will be at 1.64V.

Mechanical Component

The emergency off button will be implemented on the rear of the vehicle. The microcontroller, 1Sheeld+, voltage regulator, op amp, Arduino, and filters will be placed under the seat compartment of the Wild Thing. The joystick will be located on the right side of the vehicle where the original joystick was located.

System Test Plan

To test the joystick for the *Wild Thing*, when the joystick is pushed forward, backward, left, or right, the *Wild Thing* control board will signal to the wheel motor to rotate in the corresponding direction. When pushed forward both wheels will spin forwards and when the joystick is pushed backwards both wheels will spin backwards. If the joystick is moved left, then the right wheel should spin forward, and the left wheel should spin backward. And vice versa when moved to the left. To test the emergency off button on the rear of the vehicle, the power to both the *Wild Thing* control board and the Arduino. Resulting in the vehicle not moving when the joystick is moved.

To test the phone app, when the game pad buttons forward, backward, left and right are pushed the vehicle should move in the corresponding directions. When the emergency off button is pushed, the *Wild Thing* control board will read the rest states which keeps the vehicle in a stopped position. The only why the vehicle can be reactivated is if it is powered off then on again. This is the only way the vehicle will exit the emergency off state.

Development Process

Described below is a rough outline of the steps we followed throughout our project.

First, we purchased the required platform. Our client specified that she wanted us to modify a *Wild Thing*. Once we had one in our possession, we began discovering how it works. By figuring out how the vehicle works, it gave us a clearer idea of what kind of circuit we were going to have to implement to modify the car the way we want. After the circuit was designed and constructed, we tested it to see how well it interfaced with the vehicle. The power supply circuit was also designed and tested around this phase of the project.

Next, we selected a microcontroller and RF interface to use with our circuit. Together these controlled the vehicle. Once selected and integrated, we designed a program for our microcontroller and interface that logged timestamped data, keeping track of the vehicles use, and communicate with a cellphone. After the circuit could communicate with a cellphone, we designed an app that allowed the cellphone to control the car and upload the logged data via Bluetooth. Note that the data logging is optional for the user.

After the modified vehicle was able to be controlled by a single joystick, log data of its use, be controlled by a cellphone, and passes final testing, we will write a guide that will help others modify their *Wild Things* to operate the same way ours does.

Performance

The team was unable to complete the project due to unexpected closure of the university, resulting to a switch to online classes. Final testing was not able to be completed either. The team assumes though that once all aspects of the car had been put together, it would have functioned with no issues. The team was able to complete a working setup that allowed the vehicle to be controlled by a single joystick and to have the 1Sheeld app control the vehicle. The team still needed to complete a printed circuit board of their circuit, to install the joystick, and to place all remaining portions of the project (circuit, 1Sheeld, and Arduino) within the vehicle.

Process Outcome

Assumptions

Many of the initial assumptions remained true throughout the project. One of the assumptions that we had made initially that was made that could cause future issues was assuming the receiving family would own an Apple device. If it was possible budget wise, the team would include a mobile phone that worked with the car to give to the family. Another assumption was that all aspects of the project would work together with no issues. This remained predominantly true other than the 1Sheeld ended up requiring changes to be made to allow for sufficient current to drive the microcontroller of the car. Another assumption made was that the voltage from the microcontroller would be sufficient enough to power the Arduino put a circuit was built using capacitors to generate a higher voltage. This circuit of capacitors was later changed out for a voltage regulator (LM2596 switching regulator). Our final assumption that no longer holds true was that we would finish on time. During the year we met all goals on time and could have finished our project on time but with the universities unexpected closure, we were unable to complete the project.

Project Management

The project remained on track throughout the year until unforeseen circumstances prevented further work from being completed. The schedule for work was followed and weekly meetings allowed for the work to be completed on time. Some parts of the project took more trouble shooting than expected but they were cleared up before it called to push deadlines back. An example was powering the wheels of the vehicle using the Arduino. It required a voltage regulator that we had to initially anticipated. This was soon resolved once implementing the voltage regulator and putting the team back on track for completion.

Milestones

Milestones Completed

TABLE 3 - SPRING MILESTONES COMPLETED

Milestones	Date Completed
Ordered and received Wild Thing	October 2, 2019
Final Budget	October 10, 2019
Functional Specification Document	October 19, 2019
Dismantlement of Wild Thing	October 27, 2019
Observed and recorded all signals from original joysticks	November 8, 2019
Demonstrated that the microcontroller can connect via Bluetooth	November 10, 2019
Worked on circuits that will help implement design	November 10, 2019

Programmed the microcontroller to deliver correct voltages and current to Wild Thing control board at correct times	November 17, 2019
Designed the power supply circuit for the car and microcontroller	November 24, 2019
Design Document	November 27, 2019
Ordered all other parts needed for the design	December 4, 2019
Shiley Winter Poster Showcase	December 6, 2019
Demonstrated that the cellphone can control the car	January 31, 2020
Demonstrated that the cellphone can upload a log from the microcontroller	February 23, 2020
Demonstrated the system works when connected to the car	February 27, 2020

Milestones Not Completed

The milestones in Table 4 were not completed because of the university switching to online classes due to Covid-19.

Table 4 - Spring Milestones Not Completed

Milestones	Scheduled Completion Date
Reassemble the car with all the additions	March 20, 2020
Completed Project	March 25, 2020
Founders Day	April 13, 2020
Shiley Showcase	April 24, 2020

Risks

The items in Table 5 are possible risks that could occur. Since the university switched to online, the team was unable to test these situations and therefore, they could cause possible problems in the future.

Table 5 - Possible Risks

Risk	Severity	Likelihood

Code Failure	Medium	Low
Battery Issues	High	Low

Code Failure

Since testing wasn't completed with the code there is a possibility of failure. Unable to have easy access to the microcontroller some added elements to the code could cause some problems. The added and untested features include an upgrade to the function of the shut-off buttons on the 1Sheeld app and a shut-off if the car goes out of range of the phone's Bluetooth. The shut-off buttons should stop the car completely and cause the car to not move at all until the parent resets the microcontroller by hitting the physical emergency shut-off switch on the back of the car. Once that switch is hit to Off and then turned back to On, the car should be able to move as it did before the emergency shut-off was pressed in the application. The other feature makes sure that the car will stop and not move if it's not in range of the phone. This way the parent will have control always and not have to worry about the child's safety if the phone somehow loses connection.

Battery Issues

Even though we have added some necessary elements to make sure that the battery will be able to power everything, there is still a possibility that something might go wrong once the battery starts to lose its charge. The issues that might arise with this weren't discovered due to an inability to complete the testing. Another possible problem with the battery, although very unlikely, is that the battery could catch fire and cause harm to the client.

Conclusion

The goal of this project was to redesign the *Wild Thing* vehicle to be accessible to children with limited motor skills. Though the project was unable to be completed due to the unexpected switch to online courses, the team completed all tasks on time up until the switch. Some of these tasks included developing the code, connecting the 1Sheeld to the Wild thing interface as well as do some low-level testing. All that remained was to do final testing and final assembly of the vehicle. Overall, the project was very successful. The team hopes to support the future of the project post-graduation.

Appendices

Appendix A – Arduino C/C++ Code

```
/*
* Accessible Vehicle for Kids: Go Team Go
* Arduino code that will mimic Wild Thing controller inputs
* Written by: Andrew Shyne 11/17/2019 - 4/1/2020
*/

#define CUSTOM_SETTINGS
#define INCLUDE_GAMEPAD_SHIELD
#define INCLUDE_DATA_LOGGER_SHIELD

#include <OneSheeld.h>

// These constants won't change. They're used to give names to the pins used:
int UpInPin4 = 4;  // Digital input pin for UP SIGNAL
int RightInPin7 = 7;  // Digital input pin for RIGHT SIGNAL
int DownInPin2 = 2;  // Digital input pin for DOWN SIGNAL
int LeftInPin12 =12;  // Digital input pin for LEFT SIGNAL

int LBOutPin = 5; // PWM output pin for left BLACK
int LGOutPin = 6; // PWM output pin for left GREEN
int RBOutPin = 9; // PWM output pin for right BLACK
int RGOutPin = 10; // PWM output pin for right GREEN

//varible that is inside of infite loop in the case of remote emergency stop
int infReset = 0;

//varible that allows the user to incrementally change the max speed of the Wild Thing
```

```
//for example: if speed setting 4 is too fast and 3 is too slow, the user will be able to adjust
//this value to dial in the perfect speed
int incSpeed = 1;

void setup() {
  //initialize serial communications at 9600 bps:
  OneSheeld.begin();
  //Logger.stop();

  pinMode(UpInPin4,INPUT_PULLUP);     //Active Low
  pinMode(RightInPin7,INPUT_PULLUP);  //Active Low
  pinMode(DownInPin2,INPUT_PULLUP);   //Active Low
  pinMode(LeftInPin12,INPUT_PULLUP);  //Active Low

  pinMode(LBOutPin,OUTPUT);
  pinMode(LGOutPin, OUTPUT);
  pinMode(RBOutPin,OUTPUT);
  pinMode(RGOutPin,OUTPUT);
}

void loop() {
  int UpInPin4 = digitalRead(4);
  int RightInPin7 = digitalRead(7);
  int DownInPin2 = digitalRead(2);
  int LeftInPin12 = digitalRead(12);

//if the phone is not connected via BlueTooth to the 1Sheeld, the car will not operate
  if(!OneSheeld.isAppConnected())
  {
    analogWrite(LBOutPin, 77); //LB gets 1.5V
    analogWrite(LGOutPin, 84); //LG gets 1.64V

    analogWrite(RBOutPin, 77); //RB gets 1.5V

    analogWrite(RGOutPin, 84); //RG gets 1.64V

  }

  else

  {


    if(GamePad.isRedPressed() || GamePad.isOrangePressed() || GamePad.isBluePressed() ||
GamePad.isGreenPressed())

    {

      analogWrite(LBOutPin, 77); //LB gets 1.5V

      analogWrite(LGOutPin, 84); //LG gets 1.64V

      analogWrite(RBOutPin, 77); //RB gets 1.5V

      analogWrite(RGOutPin, 84); //RG gets 1.64V


      infReset = 0;


      while(infReset = 0)

      {

        infReset = 0;

      }

    }

    else if(GamePad.isDownPressed())

    {

     analogWrite(LBOutPin, 137); //LB gets 2.68V

     analogWrite(LGOutPin, 0*incSpeed); //LG gets 0V

     analogWrite(RBOutPin, 137); //RB gets 2.68V

     analogWrite(RGOutPin, 153*incSpeed); //RG gets 3V


     Logger.stop();
     OneSheeld.delay(2);
     Logger.start("Go Baby Go Log",true);

     Logger.add("FORWARD"," ");
     Logger.log();
    }
    else if(GamePad.isLeftPressed())
    {
     analogWrite(LBOutPin, 137); //LB gets 2.68V
     analogWrite(LGOutPin, 0*incSpeed); //LG gets 0V
     analogWrite(RBOutPin, 137); //RB gets 2.68V
     analogWrite(RGOutPin, 0*incSpeed); //RG gets 0V

     Logger.stop();
     OneSheeld.delay(2);
     Logger.start("Go Baby Go Log",true);

     Logger.add("RIGHT"," ");
     Logger.log();
    }
    else if(GamePad.isUpPressed())
    {
     analogWrite(LBOutPin, 137); //LB gets 2.68V
     analogWrite(LGOutPin, 153*incSpeed); //LG gets 3V
     analogWrite(RBOutPin, 137); //RB gets 2.68V
     analogWrite(RGOutPin, 0*incSpeed); //RG gets 0V

     Logger.stop();
     OneSheeld.delay(2);
     Logger.start("Go Baby Go Log",true);
     Logger.add("DOWN"," ");
     Logger.log();
    }
    else if(GamePad.isRightPressed())
    {
     analogWrite(LBOutPin, 137); //LB gets 2.68V
     analogWrite(LGOutPin, 153*incSpeed); //LG gets 3V
     analogWrite(RBOutPin, 137); //RB gets 2.68V
     analogWrite(RGOutPin, 153*incSpeed); //RG gets 3V

     Logger.stop();
     OneSheeld.delay(2);
     Logger.start("Go Baby Go Log",true);

     Logger.add("LEFT"," ");
     Logger.log();
    }
    else if(UpInPin4 == LOW)
    {
     analogWrite(LBOutPin, 137); //LB gets 2.68V
     analogWrite(LGOutPin, 0*incSpeed); //LG gets 0V
     analogWrite(RBOutPin, 137); //RB gets 2.68V
     analogWrite(RGOutPin, 153*incSpeed); //RG gets 3V

     Logger.stop();
     OneSheeld.delay(2);
     Logger.start("Go Baby Go Log",true);

     Logger.add("FORWARD"," ");
     Logger.log();

    }
    else if(RightInPin7 == LOW)
    {
     analogWrite(LBOutPin, 137); //LB gets 2.68V
     analogWrite(LGOutPin, 0*incSpeed); //LG gets 0V
     analogWrite(RBOutPin, 137); //RB gets 2.68V
     analogWrite(RGOutPin, 0*incSpeed); //RG gets 0V

     Logger.stop();
     OneSheeld.delay(2);
     Logger.start("Go Baby Go Log",true);

     Logger.add("RIGHT"," ");
     Logger.log();
    }
    else if(DownInPin2 == LOW)
    {
     analogWrite(LBOutPin, 137); //LB gets 2.68V
     analogWrite(LGOutPin, 153*incSpeed); //LG gets 3V
     analogWrite(RBOutPin, 137); //RB gets 2.68V
     analogWrite(RGOutPin, 0*incSpeed); //RG gets 0V

     Logger.stop();
     OneSheeld.delay(2);
     Logger.start("Go Baby Go Log",true);

     Logger.add("DOWN"," ");
     Logger.log();
```

```
    }
    else if(LeftInPin12 == LOW)
    {
      analogWrite(LBOutPin, 137); //LB gets 2.68V
      analogWrite(LGOutPin, 153*incSpeed); //LG gets 3V
      analogWrite(RBOutPin, 137); //RB gets 2.68V
      analogWrite(RGOutPin, 153*incSpeed); //RG gets 3V

      Logger.stop();
      OneSheeld.delay(2);
      Logger.start("Go Baby Go Log",true);

      Logger.add("LEFT"," ");
      Logger.log();
    }
    else
    {
       analogWrite(LBOutPin, 77); //LB gets 1.5V
       analogWrite(LGOutPin, 84); //LG gets 1.64V
       analogWrite(RBOutPin, 77); //RB gets 1.5V
       analogWrite(RGOutPin, 84); //RG gets 1.64V
    }
  }

  // wait 2 milliseconds before the next loop for the analog-to-digital
  // converter to settle after the last reading:
  delay(2);
}
```