**ELEC 291 Section 20C**

**Project 1**

**Turtle-2WD Mobile Platform: 3-Function Project**

Lab section: L2C Group #: G1-L2C Group’s Lab-Bench #s: 3C and 2C

|  |  |
| --- | --- |
| TEAM MEMBERS | STUDENT NUMBER |
| Pietr Crandall | 46939147 |
| Navjashan Singh | 29695153 |
| Diya Ren | 53483153 |
| Andrew Dombowsky | 31176143 |
| Aaron So | 17150137 |
| Nathalie Janssen | 33377145 |

***Contribution summary:***

|  |  |  |
| --- | --- | --- |
| NAME | CONTRIBUTIONS SUMMARY | CONTRIBUTION PERCENTAGE |
| Pietr Crandall | -assembly -fritzing | 16.67 |
| Navjashan Singh | -hall effect sensor code -flower functionality | 16.67 |
| Diya Ren | -optical sensor code-project leader | 16.67 |
| Andrew Dombowsky | -soldering -progress report -lab report | 16.67 |
| Aaron So | -servo functionality -range finder code | 16.67 |
| Nathalie Janssen | -motion control code -message code | 16.67 |

**Introduction and motivations**

The objective of this project was to build a multi-functional robot that operated autonomously when switched on. The first functionality of the robot was to navigate a maze. To accomplish this it must travel forward in a straight line as fast as possible and will gradually slow down as it approaches a wall. It will stop before it makes contact with the wall and then it will determine whether there is more space to either the left or right of itself and then continue in that direction. The second function of the robot is to follow a line of electrical tape. Additionally to the functions written above we have also included a functionality to draw a flower as well as a separate one to write the message “Hi!”. The ability to switch functionalities will accomplished by including two switches on the robot.

**Project Description**

**Principle Functionality 1**

For principle functionality 1, the robot moves at full speed forward in a straight line until it detects an object within 90cm. At this point it gradually slows as it continues to approach the object and stops completely before it hits the object. Then it uses the ultrasonic range sensor mounted on a servo motor (as described below) to scan left and right and determine which direction allows for a greater range of motion by comparing the sums of the readings taken from 0-90 degrees (right side) and 90-180 degrees (left side). Having determined a best direction, the robot executes a 90 degree pivot turn in that direction and begins its forward motion at full speed until it again comes within 90cm of an object in its path. Any forward motion at full speed is also controlled by the hall effect sensors (described in detail below) which ensure that the robot moves in a straight line.

1. MOTOR SHIELD:

A motor shield was added to our arduino in order to control our motors driving the wheels as well as allow us to draw power externally from batteries. This allowed us to work wirelessly and freed our robot from the limitations of being attached via a usb cable. Now it could move freely without fear of getting obstructed or getting tangled because of the wiring. In addition our robot was now powered by 5 AA batteries which provided much more power than the usb cable. This increase in power had to be compensated for in our software because the motors were now running at much higher speeds and as a result our turns needed to be recalibrated as well. In earlier testing using solely usb power we noticed that in some cases our motors were so weak due to lack of power that they appeared to not be moving at all. We compensated by greatly increasing the values for speed written to the motors, but now these changes were reverted because lack of power was no longer an issue. The motor shield reserves pins 4-7 for motor control so we had to pin plan a bit more carefully as we now had a smaller amount of pins to work with. Using the motor shield, 3 breadboards and plentiful amount of tape to attach various components and tidy up the wiring we were able to get everything up and running with a single arduino.

2.SERVO MOTOR:

We used two servo motors for this project. This first was mounted to the top front of the robot and attached to this was a miniature breadboard onto which was added an HC\_SR04 Ultrasonic Range Finder sensor as well as an LM35 temperature sensor. This was done so that we could rotate the Range finder so it could take the necessary readings. A second servo was added to the right hand side of the robot so we could attach a pen and control the position of it enabling our additional functionalities. Both servos are controlled using the standard servo library available for Arduinos.

3.ULTRASONIC RANGE FINDER SENSOR(temperature sensor):

We used an HC\_SR04 ultrasonic sensor in combination with a LM35 temperature sensor to measure the distance from the robot to an obstacle. The addition of the temperature sensor allowed for improved accuracy of distance readings because it allowed us to compensate in our calculations for variations in temperature. The distance readings from the ultrasonic sensor were then used to detect objects blocking the path of motion, and by adding threshold values we could tell the robot to gradually slow or stop when it came within a certain distance from an object.

4.HALL EFFECT SENSOR (US1881):

When we run our robot just using the DC motors, then one of our motor rotates a bit faster than the other which does not allow the robot to move in a straight line. In order to tackle this, we used hall effect sensors and attached them on both sides of our robot close to the tires. Initially we were a bit confused as to how we could use just a digital reading to make our robot go straight. But after some brainstorming and with discussions in the group, we thought of placing a magnet in the tire and a hall effect sensor close to it which could compute the time of rotation of a tire using the millis() function. In order to use the hall effect sensors, we attached them as the first pin with the voltage(5V) on arduino, second pin to the ground and the third pin to a digital pin. In order to identify which pin is which, we made sure that the right-most pin on the flat side of the hall effect sensor is pin 1 and other pins follow the corresponding pin order. We also connected a pull up resistor of 10k ohms between pin 1 and pin 3. After completing the circuitry for the hall effect sensor, we were not able to figure that how to read values from the hall effect sensor and we were getting the incorrect values. In order to tackle this problem, we tried a lot of possibilities and at the end, we found that the mode of the digital pin in input mode and we have to do a digitalRead() in order to read values from the hall effect sensors. We had our hall effect sensors working properly and then we reached the stage of attaching them on our robot. We connected the magnets on the inner side of our tires using electrical tape so that the magnets don’t come in contact with the external forces. For hall effect sensors, we also used electrical tape in order to make sure that the different pins don’t come in contact with each other. We attached the hall effect sensors on both sides between the tires and robot using a duct tape. We then ran the robot and computed the time taken for a single rotation by both tires as the hall effect sensor detects when a magnet is close to it. We figured out that one of our hall effect sensor in reading incorrect values and got it replaced. In the last phase, we had two hall effects connected on our robot which were working properly. For the coding segment of the hall effect sensors, we used millis() function to check the time taken by both the tires for rotation and compared both the timings. In order to get the correct readings, we used the technique of stalling and debouncing with both of our tires. We took a lot of readings of timings using the hall effect sensors with the robot running on the ground. We found out that the correct readings were between 400 milliseconds to 600 milliseconds. So we set a condition which checks that the timings has to be between this range and neglects the garbage values. We then compares both the readings and check if the difference between these values is greater than 100 milliseconds. If it is so, then we decrease the speed of the tire which is getting less time of rotation by 1 unit of digitalWrite() so that both the tires rotate at the same time and the robot moves in straight direction. We keep on calling our hall effect sensor function from the forward function to make sure that the robot is moving in a straight direction forward and we found that our hall effect sensors worked properly.

**Principle Functionality 2**

For principle functionality 2, the robot tracks and moves along a darker line on a lighter background. The implementation of the optical sensors to detect and follow the line, is described in detail below. Other modifications to the code included writing new turn functions which turned gradually and could stop at any time instead of completing a full 90 degree rotation as for functionality 1.

OPTICAL SENSOR:

In order to achieve the functionality 2, for hardware part, at the beginning we used half bread to attached theses optical sensors, and then we found out that if we use mini breadboard the value the sensor received would be more accurate. So we decided to attached them to a mini breadboard. After that, we taped the mini breadboard to the shelf in the front, then the assemble work of optical sensors were done.

For software part, at first we just simply use the center sensor to detect the black tape, and then used the sensors on each side to check whether the value it detected is for brighter area or not. Based on this idea, we generated some code to test this functionality. Sooner we found out that, although our robot moved totally fine on straight black tape, sometime it can not decide where to go when it reached intersection, and all it can do is to start spinning around. Then we start to debug. We noticed that due to we were using mini breadboard, the distance between two sensors was not wide enough. Although the distance is good enough for straight line, when it comes to intersection, since we usually overlap two length of tapes in order to make a intersection and so that the black tape cover range becomes wider than simply straight line, then the distance between two sensors is now not enough to let the robot to make a right decision. In order to avoid this problem showing up again, we decided to modify our code and strengthen the preconditions. At the beginning, we just simply checked the three sensor separately, and after we nailed the problem, we decided to modified the preconditions of the code in order to make sure the robot checked and compared the every two sensors’ value consistently in order to make sure it will get lost on wider black tape. For instance, when the robot reaches the intersection which is the easiest place to get lost before, the left optical sensor and center optical may all detected the black tape, under this circumstances, our modified code should make the robot gradually turn left and stop turning until the left sensor is not detecting black anymore, and then keep moving. This modification make sure our robot go through the intersection without hesitation and just keep checking and comparing until it passes the intersection safely and smoothly.

**Additional Functionality**

For our additional functionality, we decided to have our robot autonomously draw two pre-determined images onto a sheet of paper. We decided to program this functionality over others as we recognized that a pre-programmed function such as writing a word would limit the amount of environmental variables that could affect the operation of the robot, such as would be present with a new sensor. If we could decide upon a path beforehand for the robot to follow, we would be more capable of refining it and ensuring it was followed accurately. We also made this decision as we were conscious of our material limitations and didn’t want to order in new parts from an external source. We decided to have our robot perform two drawings. One of a flower, formed by a series of overlapping semicircles, and one of the phrase “Hi!”. We wanted to have two drawings to ensure a more diverse range of functionality for our robot. To provide our platform a means of writing the message, we attached a servo motor which, along with a thick marker, would rotate the marker down onto the paper when needed and lift it up and off when not needed, allowing us to run all three of our functionalities together without and external changes. We placed this combination next to the platform’s right wheel. This placement ensured more stable control of the marker during pivots, as it limited the path of the marker’s tip compared to mounting it at the back or front of the platform where the tip would travel much more during turns. Finally after completing its drawing, the robot will turn and drive off a suitable distance from the image and stop itself, to allow us or any other user to view the image and shut off the platform comfortably. For our flower drawing we had our robot complete its circles by actuating its wheels for a certain time period before stopping, moving to a new location to be the centre of the next circle, and rotating again. We used this method of drawing to ensure that each circle would be the same circumference and that tweaking the period of rotation to change the drawing would be easy to do. Examples of both draw functions can be seen in appendix D.

**References and bibliography**

Provide any relevant references. Also include the list and description of the files submitted for this lab (including code and Fritzing breadboard schematics)

Motor Shield wiki: [http://www.dfrobot.com/wiki/index.php?title=Arduino\_Motor\_Shield\_(L298N)\_(SKU:DRI0009)](http://www.dfrobot.com/wiki/index.php?title=Arduino_Motor_Shield_%28L298N%29_%28SKU:DRI0009%29)

Motor Shield schematic:

[ArduinoL298ShieldSch.pdf](https://connect.ubc.ca/bbcswebdav/pid-3136872-dt-content-rid-14227042_1/courses/SIS.UBC.ELEC.291.20C.2015W2.59753/ELEC291_15W2/Project1/291_project1_Lab_images/ArduinoL298ShieldSch.pdf)

Mobile Platform Assembly Manual:

[2WDTurtleAssemblyManual.pdf](https://connect.ubc.ca/bbcswebdav/pid-3136872-dt-content-rid-14227042_1/courses/SIS.UBC.ELEC.291.20C.2015W2.59753/ELEC291_15W2/Project1/291_project1_Lab_images/2WDTurtleAssemblyManual.pdf)

Motor Shield's motor controller datasheet:

[L298N\_datasheet.pdf](https://connect.ubc.ca/bbcswebdav/pid-3136872-dt-content-rid-14227042_1/courses/SIS.UBC.ELEC.291.20C.2015W2.59753/ELEC291_15W2/Project1/291_project1_Lab_images/L298N_datasheet.pdf)

Arduino Reference:

<http://arduino.cc/en/Reference/HomePage>

Temperature sensor IC datasheet:

[LM35\_datasheet.pdf](https://connect.ubc.ca/bbcswebdav/pid-3136872-dt-content-rid-14227042_1/courses/SIS.UBC.ELEC.291.20C.2015W2.59753/ELEC291_15W2/Project1/291_project1_Lab_images/LM35_datasheet.pdf)

HC-SR04 datasheet 1:

[HC\_SR04\_1.pdf](https://connect.ubc.ca/bbcswebdav/pid-3136872-dt-content-rid-14227042_1/courses/SIS.UBC.ELEC.291.20C.2015W2.59753/ELEC291_15W2/Project1/291_project1_Lab_images/HC_SR04_1.pdf)

HC-SR04 datasheet 2:

[HC-SR04\_Manual.pdf](https://connect.ubc.ca/bbcswebdav/pid-3136872-dt-content-rid-14227042_1/courses/SIS.UBC.ELEC.291.20C.2015W2.59753/ELEC291_15W2/Project1/291_project1_Lab_images/HC-SR04_Manual.pdf)

Reflective Optical Sensor datasheet:

[tcrt5000\_datasheet.pdf](https://connect.ubc.ca/bbcswebdav/pid-3136872-dt-content-rid-14227042_1/courses/SIS.UBC.ELEC.291.20C.2015W2.59753/ELEC291_15W2/Project1/291_project1_Lab_images/tcrt5000_datasheet.pdf)

Hall Effect Sensor datasheet:

[HallEffect\_datasheet.pdf](https://connect.ubc.ca/bbcswebdav/pid-3136872-dt-content-rid-14227042_1/courses/SIS.UBC.ELEC.291.20C.2015W2.59753/ELEC291_15W2/Project1/291_project1_Lab_images/US5881_rev007_datasheet.pdf)

Hall Effect Sensor Tutorial (note that a different hall effect sensor is used here): <http://playground.arduino.cc/Code/HallEffect>

**Appendix A – Robot pictures**

12822997_10206803136131386_283384245_o.jpg

Photo 1: Robot Side View

12823042_10206803144131586_1553035557_o.jpg

Photo 2: Robot Top View

12789675_10206803149371717_1775034082_o.jpg

Photo 3: Optical Sensor mounting

11990198_10206803170332241_1008766775_o.jpg

Photo 4: Power Switch Connections

939380_10206803203453069_41475341_o.jpg

Photo 5: Temperature and Range Finding Sensor Mountings

12810051_10206803177292415_1658454638_o.jpg

Photo 6: Hall Effect Sensor and Magnet Mounting

**Appendix B - Code**

/\*

Based on switch settings, chooses between 4 possible functionalities:

1. Moves through a course at full speed until it senses an obstacle in its path.

As it approaches obstacle, robot slows gradually until it comes to a complete stop

about 15cm away. Then it scans left and right to determine the appropriate next path,

and turns in that direction. Finally it continues it's forward motion at full speed.

While in forward motion, hall effect sensors are used to measure frequency of rotation

of the wheels and adjust speed to account for any discrepencies.

2. Tracks a dark line on a lighter background using optical sensors. If robot happens to

lose track of line, or line comes to an end, robot will turn counterclockwise until it

senses line again.

3. Additional servo lifts and lowers marker, which is then used to write a short message.

Message reads ' Hi! '.

4. Additional servo lifts and lowers marker, which is then used to draw a flower pattern

composed of semi-circles.

Last modified: Mar 8, 2016

by Group 1 (Team 2C/3C)

\*/

// Include servo library

#include <Servo.h>

// Initialize digital I/O pins (currently mostly arbitrary values)

const int servoPin = 8;

const int servo2Pin = 9;

const int echoPin = 13;

const int trigPin = 12;

const int hall1Pin = 11;

const int hall2Pin = 10;

//change switch pins!!!

const int switch1Pin = 2;

const int switch2Pin = 3;

// Initialize analog input pins

const int tempPin = 0;

const int leftSensor = 1;

const int centreSensor = 2;

const int rightSensor = 3;

// Initialize motor pins

const int E1 = 5;

const int M1 = 4;

const int E2 = 6;

const int M2 = 7;

// Initialize servo object

Servo mainServo;

Servo drawServo;

// Declare variables

int turn; //direction of longest path

int value; //speed of motors

int leftRead; //left optical sensor

int centreRead; //centre optical sensor

int rightRead; //right optical sensor

int switch1; //state of switch for principle function 1 and 2

int switch2; //state of switch for additional functionality

float tempC; //temperature of room

float tempVolt; //raw temp sensor data

float distance; //calculated distance

float speedOfSound; //calculated speed of sound

unsigned long interval; //interval for range sensor

const unsigned long maxDuration = 38000;

//variables used for checking hall effect sensors and adjusting speed of individual motors

int valueLeft = 255;

int valueRight = 255;

int testLeft = HIGH;

int testRight = HIGH;

long rotationLeft = 0;

long rotationRight = 0;

int left;

int right;

int speedDifference1;

int speedDifference2;

long firstMillis1;

long secondMillis1;

long firstMillis2;

long secondMillis2;

//variables uses for additional funcitonality

int totalMillis = 0;

int forwardTotal = 0;

int circleMillis2 = 0;

int circleMillis1 = 0;

int forwardMillis1;

int forwardMillis2 = 0;

int stopForward = 200;

int startTime = 0;

int endTime = 16000;

void setup() {

//Set up servo

mainServo.attach(servoPin);

mainServo.write(90); //Set servo to midpoint

// Set up servo for additional funtionality

drawServo.attach(servo2Pin);

drawServo.write(0);//Set servo to lift marker from ground

//Set up serial monitor

Serial.begin(9600);

//Set motor pins as outputs

pinMode(M1, OUTPUT);

pinMode(M2, OUTPUT);

// Set range finder pins to desired mode

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

//Set switch pins as inputs

pinMode(switch1Pin, INPUT);

pinMode(switch2Pin, INPUT);

//Set hall effect sensor pins as inputs

pinMode(hall1Pin, INPUT);

pinMode(hall2Pin, INPUT);

delay(1000);

}

void loop() {

//Read switch states

switch1 = digitalRead(switch1Pin);

switch2 = digitalRead(switch2Pin);

if (switch2 == 1 && switch1 == 1) {

//additional functionality (draw message: 'Hi!')

drawH();

drawi();

drawExclamation();

delay(10000);

}

else if (switch2 == 1 && switch1 == 0) {

//additional functionality (draw flower design)

drawCircles();

}

else if (switch2 == 0 && switch1 == 0) {

//principle functionality #1

rangeFinder();

}

else {

//principle functionality #2

lineTracker();

}

}

//executes primary functionality 1: sensing and navigating around walls and obsticals

void rangeFinder() {

//infinite loop to stay in this functionality

//(in order to switch functionality turn robot off and adjust switches)

while (1) {

delay(100);

// distance to next object read using readRange() function

distance = readRange();

Serial.println("functionality 1");

Serial.println(distance);

// while the distance to an object is greater than 90cm, move forward

if (distance > 90)

forward();

// if the distance to an object is less than 15cm, stop robot

else if (distance < 25) {

stopMotors();

delay(200);

// turn servo to use range finder to scan left and right

turn = scan();

//if left has more space, turn left

if (turn == 0) {

turnLeft();

//Serial.println("turning left");

}

//if right has more space, turn right

else {

turnRight();

//Serial.println("turning right");

}

}

// Otherwise if distance to object is between 15 and 40cm, slow robot down

else

slow(distance);

}

}

//execute principle function 2: tracking a line with the robot

void lineTracker() {

//infinite loop to stay in this functionality

while (1) {

// Reads values from the optical sensors

leftRead = analogRead(leftSensor);

centreRead = analogRead(centreSensor);

rightRead = analogRead(rightSensor);

// If centre sensor is reading black, move forwards

if ((centreRead > 50 && centreRead < 300) /\*&& (leftRead > 300) && (rightRead > 300)\*/)

{

opticalForward();

}

/\*

// If all sensors read black, reverse

else if ((centreRead > 50 && centreRead < 300) && (leftRead > 50 && leftRead < 300) && (rightRead > 50 && rightRead < 300)) {

for (value = 0 ; value <= 200; value += 5)

{

digitalWrite(M1, LOW);

digitalWrite(M2, LOW);

analogWrite(E1, value); //PWM Speed Control

analogWrite(E2, value); //PWM Speed Control

}

}

\*/

// If left sensor is reading black, turn left gradually

else if (/\*(centreRead > 300 ) &&\*/ (leftRead > 50 && leftRead < 300)/\* && (rightRead > 300)\*/) {

opticalLeft();

}

// If right sensor is reading black, turn right gradually

else if (/\*(centreRead > 300 ) && (leftRead > 300) && \*/(rightRead > 50 && rightRead < 300)) {

opticalRight();

}

// Otherwise (if no sensors read black), turn left until sensors find a line again

else {

opticalLeft();

}

}

}

// Turns both motors on to move forwards for principle functionality #2

void opticalForward() {

digitalWrite(M1, HIGH);

digitalWrite(M2, HIGH);

analogWrite(E1, 255); //PWM Speed Control

analogWrite(E2, 255); //PWM Speed Control

}

// Pivots robot counterclockwise until either centre or right optical sensor detects black line again

void opticalLeft() {

value = 0; //set speed to 0 initially

float left1 = analogRead(centreSensor); //read centre sensor

//execute loop while centre sensor reads white

while (!(left1 > 50 && left1 < 300)) {

//if centre sensor reads black, set left1 to read centre sensor in order to exit loop

if (analogRead(centreSensor) > 50 && analogRead(centreSensor) < 300)

left1 = analogRead(centreSensor);

//if right sensor reads black, set left1 to read right sensor in order to exit loop

if (analogRead(rightSensor) > 50 && analogRead(rightSensor) < 300)

left1 = analogRead(rightSensor);

//turn motors to pivot left

digitalWrite(M1, HIGH);

digitalWrite(M2, LOW);

analogWrite(E1, value); //PWM Speed Control

analogWrite(E2, value); //PWM Speed Control

//gradually increase speed of turn

if (value <= 100) {

value += 5;

}

}

}

// Pivots robot clockwise until either centre or left optical sensor detects black line again

void opticalRight() {

value = 0; //set speed to 0 initially

float right1 = analogRead(centreSensor); //read centre sensor

//execute loop while centre sensor reads white

while (!(right1 > 50 && right1 < 300)) {

//if centre sensor reads black, set left1 to read centre sensor in order to exit loop

if (analogRead(centreSensor) > 50 && analogRead(centreSensor) < 300)

right1 = analogRead(centreSensor);

//if left sensor reads black, set left1 to read right sensor in order to exit loop

if (analogRead(leftSensor) > 50 && analogRead(leftSensor) < 300)

right1 = analogRead(leftSensor);

//turn motors to pivot right

digitalWrite(M1, LOW);

digitalWrite(M2, HIGH);

analogWrite(E1, value); //PWM Speed Control

analogWrite(E2, value); //PWM Speed Control

//gradually increase speed of turn

if (value <= 100) {

value += 5;

}

}

}

//slows down motors gradually as it approaches an object (based on range finder data)

void slow(float dist) {

//full speed at 90cm away and stop at 15cm away

value = 255 - (3.3 \* (95 - dist));

//if distance is less than 15 and value is negative, make value 0

if (value < 0)

value = 0;

digitalWrite(M1, HIGH);

digitalWrite(M2, HIGH);

analogWrite(E1, value); //PWM Speed Control

analogWrite(E2, value); //PWM Speed Control

delay(30);

}

//turns both motors on and to high, causing robot to move forward

void forward() {

digitalWrite(M1, HIGH);

digitalWrite(M2, HIGH);

analogWrite(E1, valueLeft); //PWM Speed Control

analogWrite(E2, valueRight); //PWM Speed Control

//Serial.println("forward");

check();

}

//turns both motors on and to low, causing robot to move forward

void reverse() {

digitalWrite(M1, LOW);

digitalWrite(M2, LOW);

analogWrite(E1, 225); //PWM Speed Control

analogWrite(E2, 225); //PWM Speed Control

delay(30);

}

//stops motors and causes robot to stop its motion

void stopMotors() {

digitalWrite(M1, HIGH);

digitalWrite(M2, HIGH);

analogWrite(E1, 0); //PWM Speed Control

analogWrite(E2, 0); //PWM Speed Control

}

// pivots robot in a 90 degree turn to the right

void turnRight() {

digitalWrite(M1, LOW);

digitalWrite(M2, HIGH);

analogWrite(E1, 120); //PWM Speed Control

analogWrite(E2, 120); //PWM Speed Control

delay(550);

//reinitialize speed to max values for forward motion

valueLeft = 255;

valueRight = 255;

}

// pivots robot in a 90 degree turn to the left

void turnLeft() {

digitalWrite(M1, HIGH);

digitalWrite(M2, LOW);

analogWrite(E1, 120); //PWM Speed Control

analogWrite(E2, 120); //PWM Speed Control

delay(550);

//reinitialize speed to max values for forward motion

valueLeft = 255;

valueRight = 255;

}

//scan right and left, returns 1 if right is longest path, and 0 if left is longest path

int scan() {

float distance1 = 0;

float distance2 = 0;

for (int pos = 90; pos >= 0; pos -= 1) // goes from 0 degrees to 90 degrees

{

mainServo.write(pos); // tell servo to go to position in variable 'pos'

distance1 = distance1 + readRange();

delay(5); // waits 50ms for the servo to reach the position

}

mainServo.write(90);

delay(1000); //stops motor for 1 second

for (int pos = 90; pos <= 180; pos += 1) // goes from 90 degrees to 180 degrees

{ // in steps of 1 degree

mainServo.write(pos); // tell servo to go to position in variable 'pos'

distance2 = distance2 + readRange();

delay(5); // waits 50ms for the servo to reach the position

}

mainServo.write(90);

if (distance1 > distance2) {

return 0;//turn right

}

else {

return 1;//turn left

}

}

//read from the temp sensor and find the temperature of the room

float readTemp() {

tempVolt = analogRead(tempPin);

tempC = ((tempVolt \* 5.0 \* 100) / 1024.0); // converts temperature voltage to degrees Celsius

return tempC;

}

//read from range sensor to find distance to objects

float readRange() {

//Set trigger pin to low and then after a delay set it to high

digitalWrite(trigPin, LOW);

delayMicroseconds(4);

digitalWrite(trigPin, HIGH);

// after delaying to ensure the trigger pin is HIGH long enough

// to activate the echo pin set the trigger pin to LOW

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

//set interval to the time to recieve a signal if it takes longer

//than 38ms then no data is expected to be recieved

interval = pulseIn(echoPin, HIGH);

delayMicroseconds(40);

// temperature read using readTemp() function

tempC = readTemp();

//calculate the speed of sound in m/s

speedOfSound = 331.5 + (0.6 \* tempC);

//find the speed of sound in cm/us

speedOfSound /= 10000;

// find distance from the sensor to the object

float dist = interval \* (speedOfSound);

dist /= 2;

return dist;

}

// Function check starts

void check() {

//Reading values from the pins

testLeft = digitalRead(10);

testRight = digitalRead(11);

//Printing values on Serial Monitor

Serial.print(testLeft);

Serial.print(" ");

Serial.println(testRight);

//Assgins maximum speed in value goes less then 255

if (valueLeft <= 200 || valueRight <= 200 ) {

valueLeft = 255;

valueRight = 255;

}

//Assigns testLeft to be value 1

testLeft = 1;

//Stalling

if (testLeft == 1) {

while (testLeft == 1) {

testLeft = digitalRead(10);

}

}

//Debouncing

if (testLeft == 0) {

while (testLeft == 0) {

testLeft = digitalRead(10);

}

//Reads first time stamp

firstMillis1 = millis();

//Waits until it completes the rotation

while (testLeft == 1) {

delay(50);

testLeft = digitalRead(10);

}

//Reads second time stamp

secondMillis1 = millis();

//Computes the time for 1 rotation

Serial.print("Left Value");

rotationLeft = secondMillis1 - firstMillis1;

left = (int)rotationLeft;

Serial.println(" ");

Serial.print(left);

}

//Stalling

if (testRight == 1) {

while (testRight == 1) {

testRight = digitalRead(11);

}

}

//Debouncing

if (testRight == 0) {

while (testRight == 0) {

testRight = digitalRead(11);

}

//Reads the second time stamp

firstMillis2 = millis();

//Waits for the rotation to complete

while (testRight == 1) {

delay(50);

testRight = digitalRead(11);

}

// Reads the second time stamp

secondMillis2 = millis();

Serial.print("Right");

// Computes the time for a rotation

rotationRight = secondMillis2 - firstMillis2;

right = (int) rotationRight;

Serial.print(" ");

Serial.println(right);

}

//Conditions to select only accurate values

if ((left <= 400 || left >= 600) || (right <= 400 || right >= 600) ) {

;

}

//Computes the value for the speed to get equal rotations

else {

if (abs(left - right) >= 100) {

if (left > right) {

valueLeft = valueLeft - 1;

}

else {

valueRight = valueRight - 1;

}

}

//Write speed and changes to serial monitor for testing

Serial.print(valueRight);

Serial.print(" CHECKS ");

Serial.println(valueLeft);

Serial.print(left);

Serial.print(" COMPARES ");

Serial.println(right);

Serial.println();

}

}

/\*

\* Code for additional functionality: Writing the message 'Hi!' and drawing a flower pattern

\*/

//draw the letter 'H'

void drawH() {

drawServo.write(90); //lower servo with pen

drawStraight(1000, 1);

drawServo.write(0); //lift pen

drawStraight(500, 0);

drawRight();

drawStraight(150, 1);

drawServo.write(90); //lower pen

drawStraight(250, 1);

drawLeft();

drawStraight(400, 1);

drawServo.write(95); //adjust pen position for drawing when reversing

drawStraight(1200, 0);

drawServo.write(0); //lift pen

//move to start position for next character

drawRight();

drawStraight(500, 1);

drawLeft();

stopMotors();

}

//Draw letter 'i'

void drawi() {

drawServo.write(90); //lower pen

delay(100); //allow time for pen to move to position before drawing line

drawStraight(500, 1);

drawServo.write(0); //lift pen

delay(100);

drawStraight(100, 1);

drawServo.write(90); //lower pen

delay(500);

drawServo.write(0); //lift pen

drawStraight(600, 0);

//move to start position for next character

drawRight();

drawStraight(200, 1);

drawLeft();

stopMotors();

}

//Draw symbol '!'

void drawExclamation() {

drawServo.write(90); //lower pen

delay(500);

drawServo.write(0); //lift pen

delay(100);

drawStraight(150, 1);

drawServo.write(90); //lower pen

delay(100);

drawStraight(850, 1);

drawServo.write(0); //lift pen

drawStraight(1000, 0);

//move robot out of the way to the right so message is clearly visible, then stop all motion

drawRight();

drawStraight(3000, 1);

drawLeft();

stopMotors();

}

/\*Draws a straight line for a given distance

input: timer - determines distance robot travels and

forOrBack - determines direction of motion, 1 is forward and 0 is backward

\*/

void drawStraight(int timer, int forOrBack) {

if (forOrBack == 1) { //forward mode

long startTime = millis(); //get the timer start time

//Serial.println(startTime);

//move the desired distance

while ((millis() - startTime ) < timer) {

//Serial.print("check time: ");

//Serial.println(millis());

drawForward();

}

}

else if (forOrBack == 0) { //backward mode

long startTime = millis(); //get the timer start time

//move the desired distance

while ((millis() - startTime) < timer) {

drawReverse();

}

}

//after desired distance has been travelled, stop motors

stopMotors();

}

//Execute a 90 degree turn to the left with right wheel stationary

// (so pen remains relatively stationary)

void drawLeft() {

digitalWrite(M1, LOW);

digitalWrite(M2, HIGH);

analogWrite(E1, 120); //PWM Speed Control

analogWrite(E2, 0); //PWM Speed Control

delay(1300);

}

//Execute a 90 degree turn to the right with right wheel stationary

// (so pen remains relatively stationary)

void drawRight() {

digitalWrite(M1, HIGH);

digitalWrite(M2, HIGH);

analogWrite(E1, 120); //PWM Speed Control

analogWrite(E2, 0); //PWM Speed Control

delay(1100);

}

//Move forward at a medium speed

void drawForward() {

digitalWrite(M1, HIGH);

digitalWrite(M2, HIGH);

analogWrite(E1, 100); //PWM Speed Control

analogWrite(E2, 100); //PWM Speed Control

}

//Move backwards at a medium speed

void drawReverse() {

digitalWrite(M1, LOW);

digitalWrite(M2, LOW);

analogWrite(E1, 100); //PWM Speed Control

analogWrite(E2, 100); //PWM Speed Control

}

//Additional functionality: Draw a flower pattern composed of a series of semi-circles

void drawCircles() {

// draw (execute loop) for 16 seconds

endTime = millis() + 16000;

while (millis() < endTime )

{

//get current time

circleMillis1 = (int)millis();

drawServo.write(90); //lower pen

//draw for the duration of a second

totalMillis = 0;

while ( totalMillis <= 1000 ) {

//execute pivot turn at medium speed, while drawing petal (semi-circle)

digitalWrite(M1, LOW);

digitalWrite(M2, HIGH);

analogWrite(E1, 200); //PWM Speed Control

analogWrite(E2, 200); //PWM Speed Control

//get time elapsed since beginning of circle pattern

circleMillis2 = (int)millis();

totalMillis = circleMillis2 - circleMillis1;

//Serial.println(totalMillis);

}

//Serial.print("Entering Forward");

drawServo.write(0); //lift pen

delay(200);

forwardMillis1 = (int)millis(); //get current time

forwardTotal = 0;

//execute forward motion for a given period of time starting at 200 milliseconds

while ( forwardTotal <= stopForward ) {

//move forwards at medium speed

digitalWrite(M1, HIGH);

digitalWrite(M2, HIGH);

analogWrite(E1, 200); //PWM Speed Control

analogWrite(E2, 200); //PWM Speed Control

//get time elapsed since beginning of forward motion

forwardMillis2 = (int)millis();

forwardTotal = forwardMillis2 - forwardMillis1;

//Serial.println(forwardTotal);

}

//Serial.print("I m done");

//slowly increase length of forward motion,

// (therby increasing distance of petals from centre of circle)

stopForward = stopForward + 20;

}

//after 16 seconds have elapsed, stop motors indefinitely

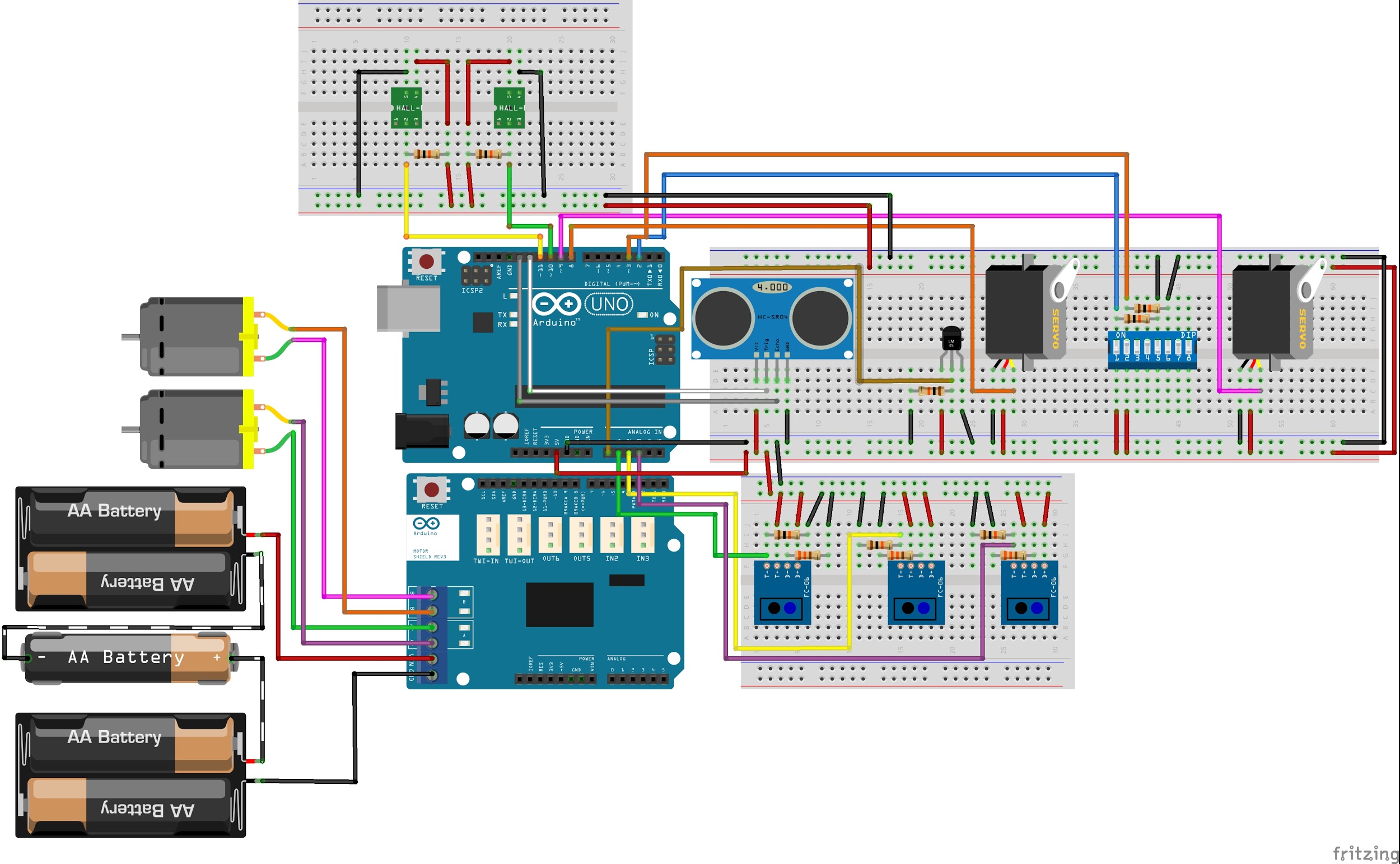
while(1){

stopMotors();

}

}

**Appendix C - Fritzing**



Fritzing Diagram of Complete Project

**Appendix D - Other**

12751764_10206803444139086_1114558273_o.jpg

Photo 7: Three outcomes of the “Hi!” function

12837406_1559889454338838_666130989_o.jpg

Photo 8: Two outcomes of the “flower” function