Aymen Mahiouz

2338768

Prof. Joel Trudeau

Comprehensive Examination

People following Robot

Introduction

This is a paper about a robot that is made to follow a person autonomously. This robot is made to follow a single person that is not in the vicinity of other people.

This is the beginning of a robot that could have many real-life applications. In fact, it could be used to replace holding something over a distance over time. For example, it could be used by elderly people who would need to carry things over a distance, or as an autonomous shopping cart.

The goal of this project is to make this robot using commonly found sensors and motors, as well as an Arduino microcontroller.

Project Description

The objective of this project is to learn how to use an Arduino, as it is the first time I am using it. I am also very unfamiliar with DC motors (and the drivers associated with them), and the PIR sensor. This is also the first time that I am using a microcontroller to make a true project (using many sensors together with hardware output). Though I have used a raspberry pi in the past, I have never seriously used an Arduino, nor coded in C/C++. To do so, I will build an autonomous robot that will follow the target person without stopping.

I did not really have any inspiration for this project. I really wanted to learn how to effectively use a PIR sensor. However, in general I believe that this interest could be fueled by many youtubers, such as Mark Rober, or Electroboom. I also have a general interest in making things, as I would like to become an engineer.

Initially, my plan was to use a PIR sensor and a continuous rotation servomotor to sweep the surroundings of the robot. I wanted to use the plotting capabilities of the Arduino, so

that it could analyze the graph of the input due to the PIR sensor due to angular position of the sensor. In fact, because the angular speed of the servo would be known, we could find the angular position of something if we knew at what time an event occurred. However, it is relevant to know that this initial plan is flawed, and that it can be greatly improved. First, because the PIR sensor only inputs 0s and 1s, there is no interest in graphing the input versus the angular position of the PIR sensor, as we only must find when the input is a 1 to know where someone is. Also, I discovered that there are two types of servomotors, and that using a fixed angle one instead of a continuous rotation one would be more accurate and efficient. This is because if I had to use a continuous rotation servo, I would have to multiply angular speed and the rotation time, to find the angle. With a fixed angle one, I can set that angle.

Implementation

For the purposes of this project, I used an Arduino Uno as the brain of the robot. It was used to connect, compile and send data to multiple hardware elements:

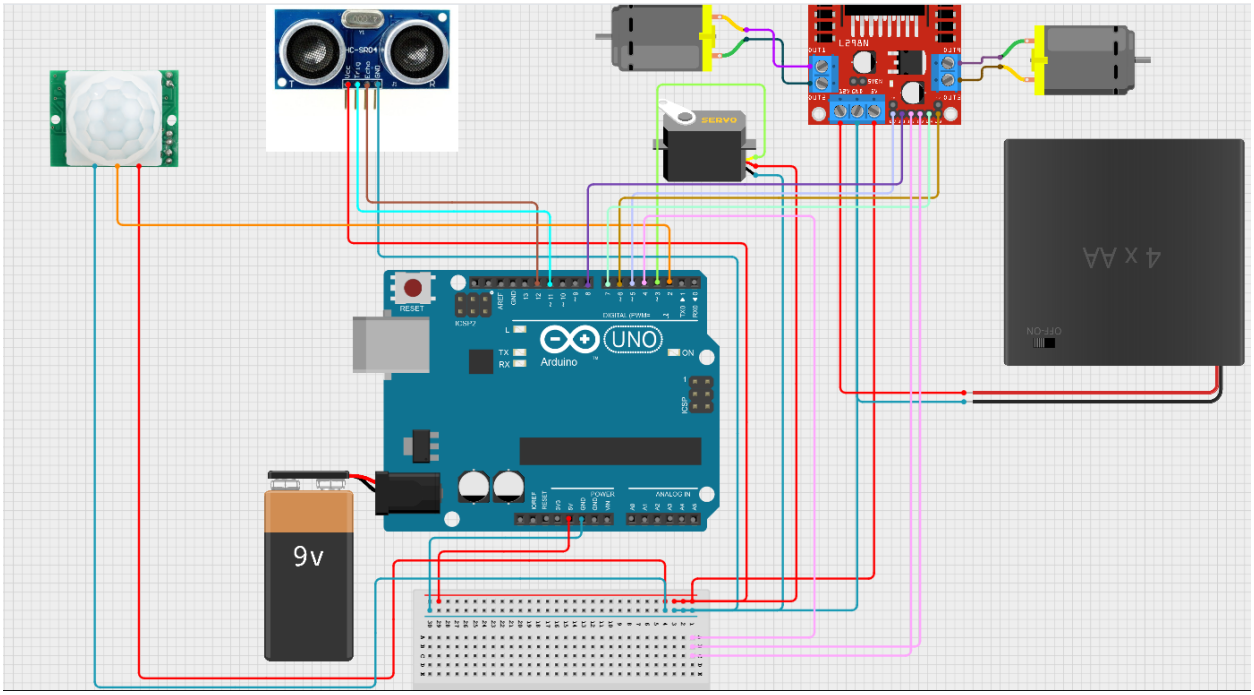
Fixed angle servomotor: The fixed angle servomotor was used to turn the PIR and ultrasound ranging sensors. It would scan from the left to the right, up to 90 degrees on each side.

PIR sensor and blinder: The passive infrared sensor was used to detect the infrared emissions emitted by the target body. It would signal when someone is in the field of view of the sensor. In fact, I put blinders on the sides of the PIR sensor so that it could only detect a body at the angles the sensor is pointing at. This is because the PIR sensor is used as home security measures, so it covers a wide angle by nature. The blinders were made from cardboard, which makes it easy to make and opaque to infrared light.

Ultrasonic range sensor: This sensor uses ultrasound to measure the distance from an object. I used it to make sure that the robot did not hit the person it was following.

Chassis: The Chassis is what supports all my robot parts. It includes the ball wheel, which allows the robot to rotate nicely on itself. This is what secures the DC motors in place, as well as the servomotor. The rest is put on place on the available space, stacked up onto each other, using instant tac.

Motors and L298N driver: The DC motors are what make the robot move. They are linked to a L298N DC motor driver module. This driver module allows the control of the motors by the Arduino. It allows the robot to control the direction of rotation of each wheel, as well as its respective angular speed.

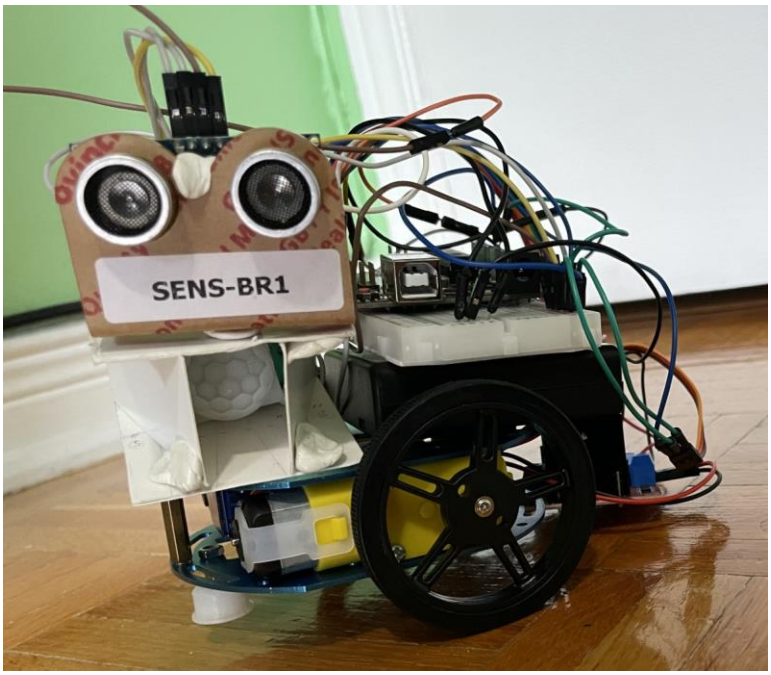


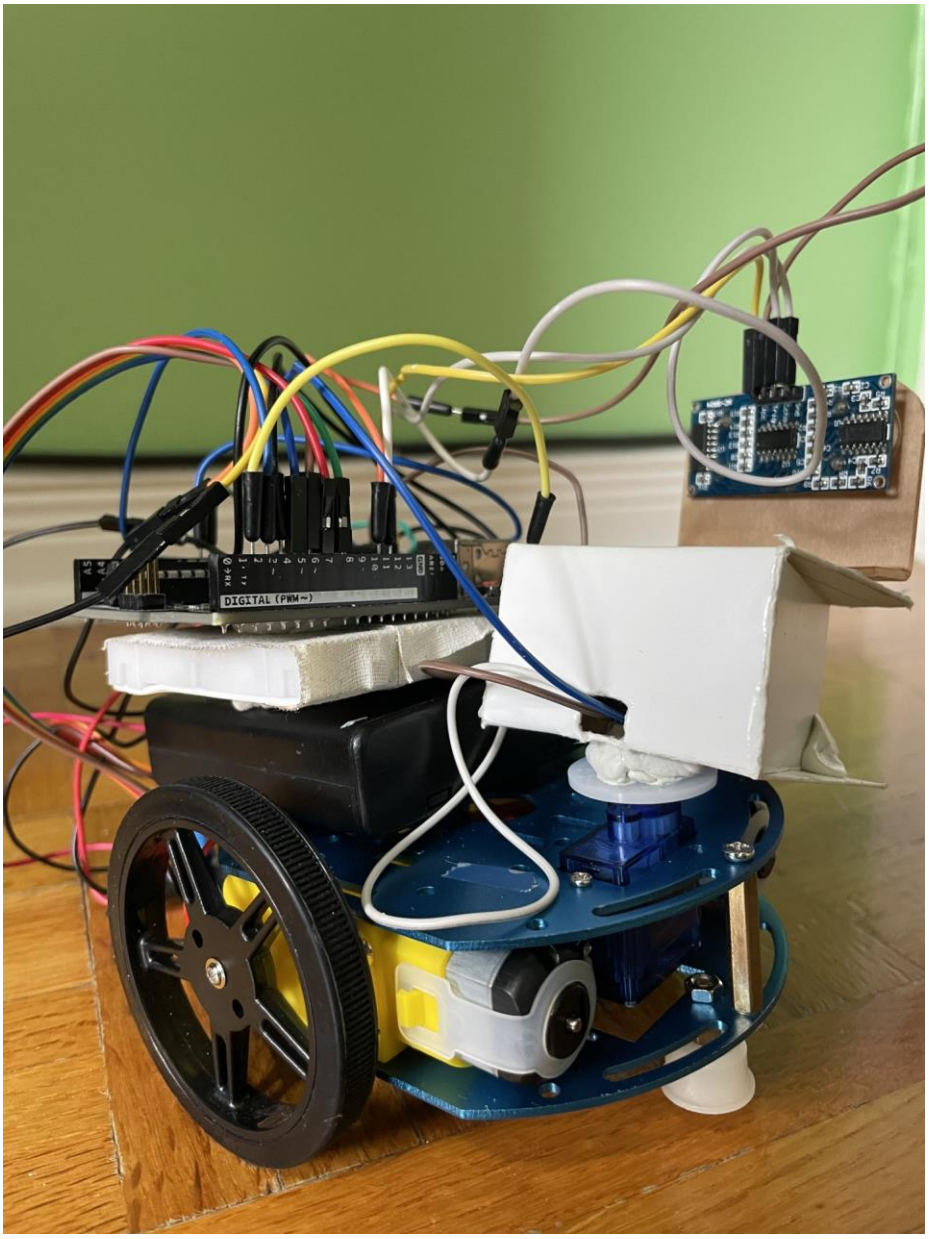
Arduino Pin	Connected to	Description
2	PIR	Input motion detection
3	Servo	Makes servo turn
4	EN2/EN3 via breadboard	Disabling general backwards motion
5	ENA	Speed control right motor
6	ENB	Speed control left motor
7	EN4	Forward control right motor
8	EN1	Forward control left motor
11	Trig (ultrasonic)	Output pulse
12	Echo (ultrasonic)	Input pulse

Required software:

Arduino IDE

Servo.h library


SENS-BR1



I wrote the code gradually. In fact, I learned how to use a component at the time, then I wrote the code associated with it in my final code. This was to make sure that every part of the code could work, as I could test it independently, and to make sure that any problem was related to another function of the robot (or not).A relevant library that I had to use was the servo.h library. In fact, I did not need any library to control the L298N motor driver (and therefore the DC motors), the PIR sensor or the ultrasonic range sensor. This library was used to define the relevant pin to use, as well as to define the angle of the servomotor.

The code logic of the robot is straightforward.

```
#include <Servo.h>
```

```
Servo sweepservo;
int PIR=2; //PIR sensor
int motor1 = 7;//right wheel motor
int backpins = 4;// backwards activation pins for both motors
int motor2 = 8;//left wheel motor
int motorcontrol1= 5;//speed right motor
int motorcontrol2= 6;//speed left motor
int trigpin=11; //ultrasonic range sensor trigger
int echopin=12; //ultrasonic range sensor echo
//
void setup() {
  Serial.begin(9600);
  sweepservo.attach(3); //servomotor
  pinMode(PIR,INPUT); //setting up PIR
  pinMode(motor1, OUTPUT); //setting up right motor
  pinMode(backpins, OUTPUT); //setting up backwards activation pins for both
motors
  pinMode(motor2, OUTPUT); // setting up left_motor
  pinMode(motorcontrol1, OUTPUT); //setting up right motor speed control
  pinMode(motorcontrol2, OUTPUT); //setting up left motor speed control
  pinMode(trigpin,OUTPUT); //setting up trigger pin
  pinMode(echopin,INPUT); //setting up echo pin
  digitalWrite(backpins, LOW); //making sure that the wheels never go backwards
  analogWrite(motorcontrol1,200); //speed that makes both wheel go the same speed
in real life
  analogWrite(motorcontrol2,160); //speed that makes both wheel go the same speed
in real life
  delay(30000); // waiting for the PIR sensor to be warmed up
}
```

To begin, the Servo.h library is imported. Then, pins associated with the different sensors and motor components are determined. Then, those pins are set as either input or outputs. It is important to note that the angular speed of both wheels are not the same in the code as I had to compensate for realistic factors that made a wheel faster than the other, if they had the same speed in the code. I wanted my robot to move as straight as possible.

```
void forward() {//makes robot go forward
  digitalWrite(motor1,  HIGH);//rigt wheel moves
  digitalWrite(motor2, HIGH); //left wheel moves
}

void left_turn() {//makes robot turn left
  digitalWrite(motor1,  HIGH);//right wheel moves
```

```
  digitalWrite(motor2, LOW); //left wheel is stationary

}

void right_turn() {//makes robot turn right
  digitalWrite(motor1,  LOW);//right wheel is stationary
  digitalWrite(motor2, HIGH);//left wheel moves

}
void stop() {//makes robot stop
  digitalWrite(motor1,  LOW);//right wheel is stationary
  digitalWrite(motor2, LOW); //left wheel is stationary

}
void trig_pulse(){//send ultrasonic pulse
  digitalWrite(trigpin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigpin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin, LOW);
}
```

I then set up the functions that would set up the motors for relevant motion modes: forward, left turn, right turn, and stopping. This is made by playing with the motor logic. Because the pins that make the wheels go backwards are set to always be on Low, I only need to manipulate the pins that allow a wheel to go forwards (either HIGH or LOW). Another function that sends a trigger ultrasonic sound wave was created. This is done by turning it on for only 10 microseconds.

```
void loop() {
  long duration;
  float distance;
  int left_sum=0;//sum of all detections on the left side
  int right_sum=0;//sum of all detections on the right side
  ////
  ///first sweep
  int AngleToState1[7][2]={{0,0},{30,0},{60,0},{90,0},{120,0},{150,0},{180,0}};//
assigns a state to each tested angle(initially zero)
  for(int i=0;i<7;i++)
  {
    int angle=i*(180/6);//turns by 30 degrees each time
    sweepservo.write(angle);
```

```
    AngleToState1[i][1]=digitalRead(PIR);//assigns to the angle value in
AngleToState the reading value(0 if not detection, 1 if detection)
    Serial.println(AngleToState1[i][1]);
    Serial.println(angle);
    delay(1300);// wait in order to reinitialize the 0 state of the sensor
  }

  for(int i=0;i<3;i++)
  {
    left_sum=left_sum+AngleToState1[i][1]; //adds the number of detections on the
left side
  }

  for(int i=4;i<7;i++)
  {
    right_sum=right_sum+AngleToState1[i][1];//adds the number of detections on
the right sie
  }

  ///second sweep
  //refer to first sweep for comments
  int
AngleToState2[7][2]={{0,0},{30,0},{60,0},{90,0},{120,0},{150,0},{180,0}};//9?
  for(int i=0;i<7;i++)
  {
    int angle=i*(180/6);
    sweepservo.write(angle);
    AngleToState2[i][1]=digitalRead(PIR);
    Serial.println(AngleToState2[i][1]);
    Serial.println(angle);
    delay(1300);
  }

  for(int i=0;i<3;i++)
  {
    left_sum=left_sum+AngleToState2[i][1];
  }

  for(int i=4;i<7;i++)
  {
    right_sum=right_sum+AngleToState2[i][1];
  }
  /////
```

```
  delay(1300);
  sweepservo.write(90);
```

The main loop starts by initializing the sum of detections on both sides of the robot to zero. The robot, for every 30 degrees swept by the PIR sensor, will either store a 0 value, or a 1 value. All of the values for the angles are 0 by default, but become 1 when something is detected at the angle. There is a delay between each time the servo changes angles to make sure that the PIR sensor comes back to a 0 state. The robot sweeps twice before making a decision to make sure that its data is more reliable. After each sweep, it adds up the detection numbers for both the right and left side of the robot.

```
if(AngleToState1[3][1]==1 && AngleToState2[3][1]==1 && right_sum==left_sum){//
the pir sensor detected a body in front of the robot
    Serial.println("straight");
```

Then, if both sweeps indicate that there is a detection in front of the robot, and that the number of total detections for the right and left side is the same, the robot will go straight.

```
if(right_sum>left_sum){//the PIR sensor detected a body to the right of the robot
    Serial.println("go right");
    right_turn();//turn on yourself to the right
    while(true){
      if (digitalRead(PIR)==1){//if the target body is detected
```

*Example using right side

If the robot has a greater number of detections on a side, it will go on that side. When it goes on the side, there is an extra step. In fact, the robot will rotate itself, in the direction of most detections, until it detects a target body in front of it. It will then go forwards.

```
        forward();
       for(int i=0;i<200;i++){//go straight for 2 seconds or stop when 5 cm from
an object
          delay(10);
          trig_pulse();//send pulse
          duration = pulseIn(echopin, HIGH);
          distance = duration * 0.0343 / 2;//use speed of sound equation to find
the distance
          Serial.println(distance);
          if(distance<5){// check if distance is larger than 5 cm
            break;
```

The following applies to when the robot goes straight, or after the robot has made its rotation. The robot will go straight, for either two seconds, or until it is 5 cm from an object.

Then, the loop starts again.

The first sensor I used is the Passive Infrared Sensor. This sensor is used to detect the movement of warm bodies. First, we have to understand that the sensor has two slots, with each slot covering a side of the sensor (say right and left). Each slot has a distinct detecting area. These slots are made from material that is sensitive to infrared radiation. When no warm body is moving in front of the sensor, both slots receive the same amount of infrared radiation, which causes no differential. When a warm body moves in front of it, it must pass through the detection area of a slot. This will make a difference in infrared radiation received in both slots, which causes a differential. When coming into the first slot, this is called a "positive differential change", and when it is leaving the second slot, it is called a "negative differential charge". This change in potential is what allows the sensor to detect movement. The "sensing" part of the sensor is placed in a hermetic metallic container, to reduce interference from the elements. In this seal is an IR-transmittive material that allows the flow of IR radiation at a certain part, whilst still protecting the sensor. To be the most effective, the sensors have a Fresnel lense, which provides a larger range to the sensor.

Knowing that the PIR sensor detects change of Infrared light in its own frame of reference, instead of leaving the sensor stationary, and making a warm body move, we can make the PIR sensor move and leave the warm body stationary. In the sensor's frame of reference, both situations generate the same result: movement of warm bodies is detected. For many angles, I made the sensor read its surroundings. In fact, if between 2 angles there would be a warm body, the movement of the sensor would trigger it.

Here is a code block that sets up the PIR sensor:

```
int PIR=2; //PIR sensor
```

This line of the code determines the pin 2 to be attributed to the PIR sensor

As well as

```
 pinMode(PIR,INPUT); //setting up PIR
```

This line of code put the pin 2 as an input

Here is a code block that is used to control the PIR sensor:

```
  AngleToState1[i][1]=digitalRead(PIR);//assigns to the angle value in AngleToState the reading value(0 if not detection, 1 if detection)
```

This line of code to assigns a certain angle a value of the state of detection of the PIR sensor. In fact, AngleToState1 is an array that stores arrays composed of an angle value and a detection state. If it is 0, there is not detection, and if it is 1, there is a detection.

As well as

```
  if (digitalRead(PIR)==1){//if the target body is detected
```

This line of code allows the robot to understand that it is facing someone, during its rotation phase, if the target is on its side. If it detects someone, the PIR sensor inputs 1 as a value, which tells the robot that it is facing the person.

The other sensor that I used is the ultrasonic range sensor. It is used to collect data that will serve to determine the distance between the sensor and the nearest surface. This sensor first sends a sound wave at a frequency that is too high for a human to perceive. It then waits for the sound wave to bounce on an object, then comes back to the sensor, where it receives the waves it sent. The time interval can then be used to determine the distance between the sensor and the object, as it corresponds to twice the time sound would take to travel to the object.

Using the relationship between time, distance and speed of sound, it is possible to determine the distance from the object. Assuming that the speed of sound in air is 0.0343 cm/s, I divided the time by two, then multiplied that number by the speed of sound to obtain the distance between the ultrasonic range sensor and a surface. I used it to make sure that it never advances if the robot is less than 5 cm from its target or any surface.

Here is a code bloc that sets up the ultrasonic range sensor:

```
 int trigpin=11; //ultrasonic range sensor trigger
int echopin=12; //ultrasonic range sensor echo
```

These lines of code determine pins 11 and 12 to be respectively as triggers and echo.

As well as

```
pinMode(trigpin,OUTPUT); //setting up trigger pin
  pinMode(echopin,INPUT); //setting up echo pin
```

These lines of code put pin 11 as an output, and pin 12 as an input.

And

```
void trig_pulse(){//send ultrasonic pulse
  digitalWrite(trigpin, LOW);
  delayMicroseconds(2);
```

```
  digitalWrite(trigpin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin, LOW);
}
```

These lines of code serve to create a function that will trigger a specific ultrasound wave. This function is used later in combination with more code to determine the distance.

A code block that uses the ultrasonic range sensor is

```
    trig_pulse();//send pulse
    duration = pulseIn(echopin, HIGH);
    distance = duration * 0.0343 / 2; //use speed of sound equation to find the distance
    Serial.println(distance);
```

These lines of code are used to determine the distance from the robot to a surface at a certain instant. First, a soundwave is sent, then it is picked up. The time elapsed between these two times is processed mathematically, using the sound in air model, to give the distance.

Results

The robot was not able to detect the location of a person that was to be followed. To be clear, success would be defined by the ability to detect, point towards, and move towards a person. The robot does not do that. It is relevant to state that the sweeping motion of the sensors, the determining of the distance from the robot to an obstacle, and the moving of the wheels of the robot are successful. However, despite a double sweep of the PIR sensor, its results are not reliable, as it often does not detect a person, or falsely detects a person at an angle where there is nobody. Most of the time, it does not detect a person, so it keeps sweeping again and again. Even if sweeps were to be successful in finding where the person is, it would still fail as the robot would have to spin on itself and detect again when it would be in front of the person. That being said, during independent testing, all the rest of the robot's functions work properly. I collected information through observation. In fact, during testing, the robot would continuously sweep in front of the person, not detecting it, or it would decide on which side the target person is(sometimes the right and other times the wrong side), turn on itself, and go in a direction that is not towards the target. Data collected through the serial terminal back up my claim. During testing, the state of the sensor (detection or not) relative to an angle would rarely be reliable.

Analysis

The initial objective was to make a robot that detects, then follows a person. The “logic chain” of the robot is broken by wrong PIR sensor readings. In fact, during testing, the robot was able to move (in the wrong direction) and to detect its distance from a surface. The malfunction of the PIR sensor makes the robot “blind”. In fact, it has no sense of correct orientation, as it is the only component that makes the robot see its target. I believe that the implementation of the PIR sensor in this project was wrong. One explanation could be that PIR sensors are not made to have blinders mounted on them. This comes from the thought that they are made to cover wide angles, so having their vision field being reduced makes them very unreliable. In fact, knowing that they have two slots of detection, maybe that each individual slot was too small to detect anything, knowing that there might be a blind spot in the middle of them. However, for this project, widening the blinders would not be a great idea, as the sensor would cover a too wide area at once, which would make it reliable to detect specific angles. Also, another hypothesis would be that, in the sensor’s frame of reference, the warm body might move too quickly, as the fixed angle servomotor turns fast. This would not give enough time for the sensor to detect any significant change in potential in the 2 slots, as it could maybe consider it as noise.

Discussion and Conclusion

Though I might not have achieved the goal to make a working robot, this project is such a win for me. I learned how to use an Arduino, used Github for the first time, soldered for the first time, learned how to use a DC motor with a microcontroller (using a motor driver), and built the first project with a chassis.

The sweeping motion of the sensors of the robot’s surroundings went well. This is because the fixed angle servomotor was reliable and well calibrated. It did not jitter.

Another thing that went well was the robot’s displacement. The logic for it was simple, as the backwards pins were always disabled (explained later), I only had to change the state of the front pin. Therefore, if both motor’s were LOW; no movement, if only one was HIGH; turn on itself, and if both were HIGH; forward motion.

The first challenge that occurred to me was when all of the DC motor wires disconnected, as they were connected to it by hot glue. I took my father’s soldering kit, for the first time, and fixed the situation.

Another issue I faced was that I did not have enough PWM pins. To remedy this, I chose to assign a same pin to the pin of the motors that are implemented to make them go backwards. In fact, in a motor, there is a pin that serves to give the state of activation of the direction of the motor (backwards and frontwards). Because i knew that my robot did not

have to go backwards, I connected both backwards pins of the two motors, and connected them to a pin in the Arduino (known a “backpins” in the code). This pin as always on the LOW state, which puts both motors’ backwards state as LOW. I later learned that the motor driver did not need a PWM pin for motor direction control. This served as an optimization lesson.

To continue, another problem I had was the speed control of the motors. In fact, due to an inequality between the motors, they had to have different speeds so that their real-life radial speed was equal, in order to make the robot go straight. Therefore, I had to specify a certain speed for each motor, which required PWM pins. In independent testing, the wheels worked very well. However, when implemented in the code, the wheels would not turn. After hours of troubleshooting, code testing and research, I learned that the Servo library disabled the pins 9 and 10. This explained why they could work during independent testing, but not when implemented in the code. I simply changed the PWM pins for the speed control, and did not use pins 9 and 10.

Another issue that I faced was that I did not have enough power to power everything. In fact, at the beginning, I did not have enough power to operate only the servomotor and the PIR sensor at the same time. Another example would be that when I plugged the Arduino to my laptop. The motors would not move. This is why my final product has 4 AA batteries, mainly for the DC motors, and a 9V battery. This powering setup made my robot work well.

To make a big picture of the project, the robot can be divided into two big parts: data collection and displacement. The data collection part is directly made by both the PIR sensor and the ultrasonic range sensor, and is facilitated by the servomotor. The displacement part is assured by the chassis, the 2 DC motors, and a motor driver. All of this is made so that the robot detects and follows a person. All the hardware setup was successful, with complete wiring and a stable structure, sticking things with instant tac. The code was tested by successfully making each component work. The result was not expected, as the PIR sensor failed to pick up on warm bodies.

If I had to do this project again, I would replace the PIR sensor with a camera. In fact, using body recognition, the robot would be able to track a person in a crowd. In fact, the PIR sensor cannot discriminate between many warm bodies. This would allow the robot to follow a single person in a crowd, even with warm, non-human bodies. Also, it would be great to find the distance between the robot and an obstacle through this same camera. This is because if this technology were to be implemented in people’s daily lives, ultrasounds would not be great to work with as they could negatively impact animals that

could hear those sounds. An extension of the project would be to add more powerful motors, and a cart to the robot, so that it could carry something.

Reference

Arduino Forum. (2013, March). *How to use servo library without disabling PWM pins 9 & 10*. Arduino Forum. https://forum.arduino.cc/t/how-to-use-servo-library-without-disabling-pwm-pins-9-10/153694

Cook, J. S. (2018, April 4). *Ultrasonic sensors: How they work (and how to use them with Arduino)*. Arrow.com. https://www.arrow.com/en/research-and-events/articles/ultrasonic-sensors-how-they-work-and-how-to-use-them-with-arduino

Electronicsfan123. (2021, June 22). *Interfacing Arduino Uno with PIR motion sensor*. Arduino Project Hub. https://projecthub.arduino.cc/electronicsfan123/interfacing-arduino-uno-with-pir-motion-sensor-593b6b

Isaac100. (2017, August 5). *Getting started with the HC-SR04 ultrasonic sensor*. Arduino Project Hub. https://projecthub.arduino.cc/Isaac100/getting-started-with-the-hc-sr04-ultrasonic-sensor-7cabe1

Lady Ada. (2014, January 28). *PIR motion sensor*. Adafruit Learning System. https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work

Acknowledgment

I would like to thank my CE mentor, Prof. Joel Trudeau, for providing me with most of the hardware I needed, such as the chassis, the DC motors, the motor driver, the ultrasonic range sensor, the Arduino, a breadboard and a servomotor. Additionally, he guided me through the project, helping me resolve some of my problems

I would like to mention the generosity of Abdulrahman Kikia, who gave me the 9V battery that would serve to power my robot.

I would like to thank my father for providing me with a soldering kit.