



Birzeit University
Faculty of Engineering and Technology
Department of Electrical and Computer Engineering Department
ENCS 4110
Computer Design Laboratory

SMART HOME AUTOMATION PROJECT

Prepared by:
Ayah Hamdan 1180808

Section:
6

Supervised by:
Dr. Ahmad Afaneh

Teacher Assistant:
Eng. Heba Qadah

May 21th 2021

Abstract

This Project is an implementation for a smart home, where by using specific sensors, automated responses will occur when needed. For instance, if the temperature increases above a limit, the fan will automatically turn on, and if there was movement in a specific area, then the door will open, also if there was a movement and no enough light was presented, then a bulb will turn on. Additionally, this project allows the user to control the responses for the devices when needed by sending notifications.

Table of Contents

Table of Figures	IV
Theory	1
Design and implementation	3
Ultrasonic sensor and the servo motor:.....	3
Figure 12: Ultrasonic sensor and the servo motor Circuit	3
Figure 13: Get Distance function Code	4
Figure 14: Door Functions Code.....	4
LDR, PIR, and the Bulb:.....	4
Figure 15: LDR, PIR, and the Bulb Circuit	5
Figure 16: get LDR and PIR values Code	5
Figure 17: LDR and PIR Code.....	5
Temperature Sensor and DC Motor:.....	6
Figure 18: Tmp36 and DC motor Circuit	6
Figure 19: Obtain Temperature value Code.....	6
Figure 20: Tmp36 Code	6
The Switch:	7
Figure 21: Switch Circuit.....	7
Figure 22: Switch Code	7
LEDs and Buzzers:	7
Figure 23: LEDs and Buzzers Circuit.....	8
Figure 24: LEDs and Buzzers Code.....	8
Remote Control:.....	8
Figure 25: IR Remote and IR Sensor Code	9
Manual Mode:.....	9
Figure 26: Manual Code 1	9
Figure 27: Manual Code 2	10
Figure 28: Manual Code 3	10
Testing.....	11
1) Automatic Mode:.....	11
Figure 29: Auto Mode Test- Serial Monitor 1	11

Figure 30: Auto Mode Test- Serial Monitor 2	11
Figure 32: Auto Mode Test- Door Test	11
Figure 31: Auto Mode Test- Fan Test.....	11
Figure 33: Auto Mode Test- Bulb Test.....	11
Figure 34: Auto Mode Test- All results 1	12
Figure 35: Auto Mode Test- All results 2	12
2) Manual Mode:.....	13
Figure 37: Manual Mode Test- Serial Monitor 2.....	13
Figure 36: Manual Mode Test- Serial Monitor 1	13
Figure 38: Manual Mode Test- Door Test	13
Figure 39: Manual Mode Test- Door Result.....	13
Figure 40: Manual Mode Test- Bulb Test.....	14
Figure 41: Manual Mode Test- Bulb Result 1	14
Figure 42: Manual Mode Test- Bulb Result 2	15
Figure 43: Manual Mode Test- DC Motor Test.....	15
Figure 44: Manual Mode Test- DC Motor Result	16
Figure 45: Manual Mode Test- Final Result.....	16
Conclusion	17
References	18
Appendix.....	19
Code for the First Arduino:.....	19
Code for the Second Arduino:	24

Table of Figures

Figure 1: Ultrasonic Sensor.....	1
Figure 2: Servo motor (TinkerCad)	1
Figure 3: Servo motor.....	1
Figure 4: LDR.....	1
Figure 5: PIR (TinkerCad)	2
Figure 6: PIR.....	2
Figure 7: TMP 36.....	2
Figure 8: DC Motor	2
Figure 9: DC Motor (TinkerCad)	2
Figure 10: IR Remote and Sensor.....	2
Figure 11: The Circuit.....	3
Figure 12: Ultrasonic sensor and the servo motor Circuit.....	3
Figure 13: Get Distance function Code.....	4
Figure 14: Door Functions Code.....	4
Figure 15: LDR, PIR, and the Bulb Circuit.....	5
Figure 16: get LDR and PIR values Code.....	5
Figure 17: LDR and PIR Code.....	5
Figure 18: Tmp36 and DC motor Circuit.....	6
Figure 19: Obtain Temperature value Code.....	6
Figure 20: Tmp36 Code.....	6
Figure 21: Switch Circuit.....	7
Figure 22: Switch Code.....	7
Figure 23: LEDs and Buzzers Circuit.....	8
Figure 24: LEDs and Buzzers Code.....	8
Figure 25: IR Remote and IR Sensor Code.....	9
Figure 26: Manual Code 1.....	9
Figure 27: Manual Code 2.....	10
Figure 28: Manual Code 3.....	10
Figure 29: Auto Mode Test- Serial Monitor 1.....	11
Figure 30: Auto Mode Test- Serial Monitor 2.....	11
Figure 31: Auto Mode Test- Fan Test.....	11
Figure 32: Auto Mode Test- Door Test.....	11
Figure 33: Auto Mode Test- Bulb Test.....	11
Figure 34: Auto Mode Test- All results 1.....	12
Figure 35: Auto Mode Test- All results 2.....	12
Figure 36: Manual Mode Test- Serial Monitor 1.....	13
Figure 37: Manual Mode Test- Serial Monitor 2.....	13
Figure 38: Manual Mode Test- Door Test.....	13

Figure 39: Manual Mode Test- Door Result.....	13
Figure 40: Manual Mode Test- Bulb Test.....	14
Figure 41: Manual Mode Test- Bulb Result 1.....	14
Figure 42: Manual Mode Test- Bulb Result 2.....	15
Figure 43: Manual Mode Test- DC Motor Test	15
Figure 44: Manual Mode Test- DC Motor Result	16
Figure 45: Manual Mode Test- Final Result.....	16

Theory

The Arduino is an open-source platform consisting of a hardware programmable circuit board aka the microcontroller and a software part or IDE used to write and upload the computer code to the physical board, and it is used for building electronics projects, where it can interact with buttons, LEDs, motors, speakers, cameras, computers and many more. [2] It has a built-in ADC (analog-to-digital converter), which converts the analog voltage (from 0-5V) into a digital value in the range of (0-1023). [6]

The Ultrasonic Sensor (Distance Sensor), sends ultrasonic waves which are sounds can't be heard by the human, and if they bounce back to the sensor, then it recognizes an object. The distance to the object can be measured from the time taken for the ultrasonic wave to bounce back, where the sound waves always travel at the same speed through the air, which is 343m/s. And it can be written in the formula: $\text{Distance} = 343 * \text{Time}$. [3] The HC-SR04 sensor as shown in Figure 1, has 4 pins: VCC, which is connected to 5V, GND, which is connected to Ground, TRIG (Trigger), which is connected to the transmitter to send the ultrasonic burst, and ECHO, which is connected to the receiver. [4]



Figure 1: Ultrasonic Sensor

The servo motor has an output shaft which can be positioned to specific angular positions by sending the servo a coded signal. It consists of 3 pins as shown in figure 2, VCC, which is connected to 5V, GND, which is connected to Ground, and the remaining pin connected to the digital pin in the Arduino. In the code, the servo library should be included, then declaring the servo object and initializing it using the `servo.attach()` function, and it has a built-in function: `servo.write(degrees)` which controls it. [5]

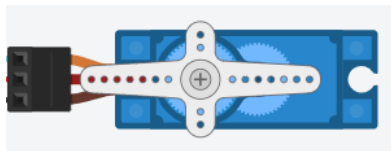


Figure 2: Servo motor (TinkerCad)



Figure 3: Servo motor

The Light Dependent Resistor – LDR, is used to detect the intensity of light or darkness in its environment. The LDR gives out an analog voltage when connected to VCC (5V), which varies in magnitude in direct proportion to the input light intensity on it. That is, the greater the intensity of light, the greater the corresponding voltage from the LDR will be. Since the LDR gives out an analog voltage, it is connected to the analog input pin on the Arduino. [6]

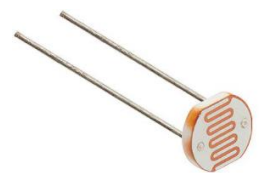
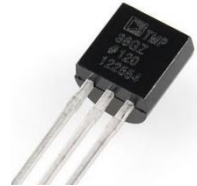


Figure 4: LDR

The Passive InfraRed Sensor (PIR), detects objects with infrared light levels using the white dome, which is a lens that expands the IR detector's field of vision. It can report if there is movement or not, but cannot detect the distance. The sensor reports a LOW signal by default,

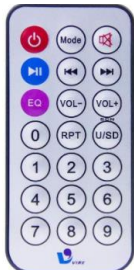
The temperature sensor (TMP36), is used to measure temperature with range between (-40°C to 125°C). the power supply for it ranges between 2.7V-5.5V, and the analog voltage reading will range regardless the supply between 0-1.75V. thus the output pin must be connected to an analog pin. The temperature can be obtained by turning the 10-bit analog reading (0-1023) into millivolts and then into temperature.[8]



The DC motor rotates at two directions in different speeds. The speed of the DC motor is controlled by controlling the input voltage to the motor and one way of doing that is by using PWM signal. The pulse width modulation (PWM) is a technique which allows to adjust the average value of the voltage that's going to the electronic device by turning on and off the power at a fast rate. The average voltage depends on the duty cycle, or the amount of time the signal is ON versus the amount of time the signal is OFF in a single period of time. Inverting the direction of the current flow through the motor results in inverse rotation, and it can be done using the H-Bridge. [9]



The IR remote is used with IR receiver to communicate with each other by transmitting and decoding a signal in the form of pulsed IR radiation. Where the Infrared radiation (IR) is a type of electromagnetic radiation invisible to the human eye because of its wavelengths range. Using signal modulation, the IR light source at the end of the remote is blinked with a specific frequency and the amount of time the signal stays high or low and the number of sent bits for each command is different for all of the IR protocols. To enable the IR remote, the IRremote Arduino library must be used. [10]



2

Design and implementation

The circuit implemented is as shown in the figure below.

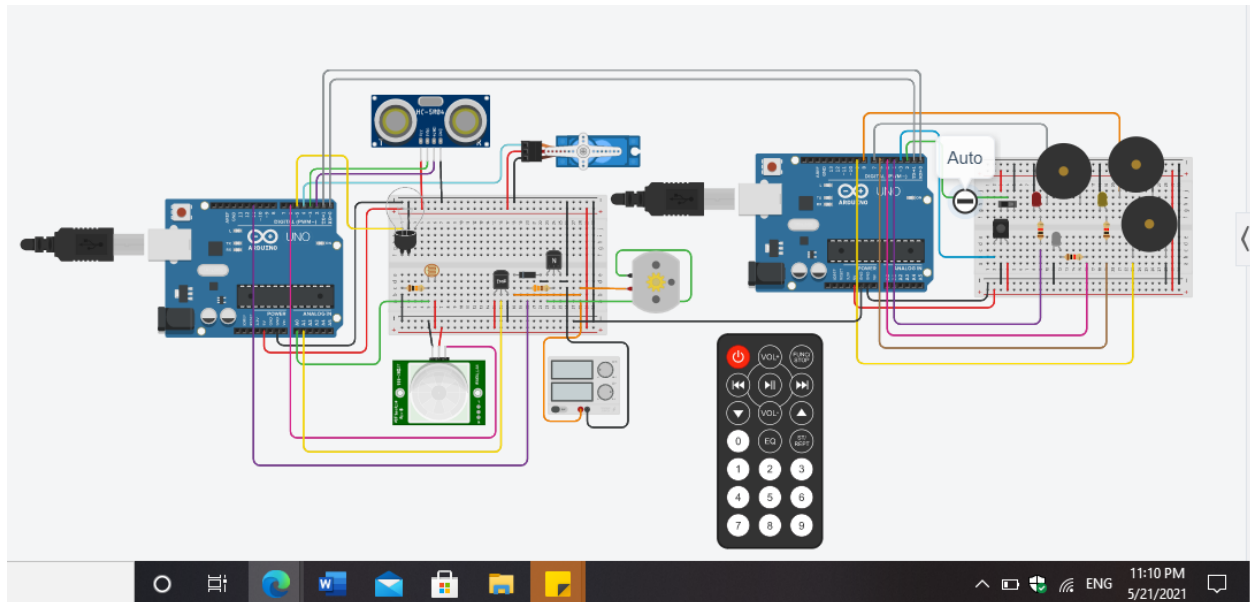


Figure 11: The Circuit

Ultrasonic sensor and the servo motor:

The connection for it is simple as shown in the figure, where the VCC and GND for both devices is connected to the 5v and GND of the Arduino respectively, and the other pins is connected to digital pins for the Arduino.

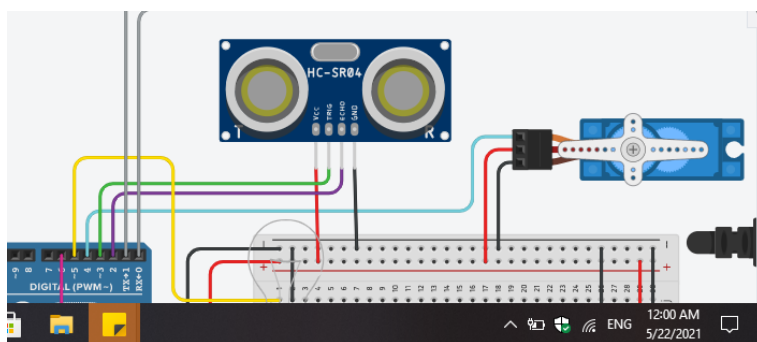


Figure 12: Ultrasonic sensor and the servo motor Circuit

First, to find the distance using the ultrasonic sensor, the following code was written, where before sending the pulses using the trigger pin, it must be deactivated to insure a clean high pulse, and the trigger pin was high for 10us, where during that time it transmitted the pulses. After that, the trigger pin was deactivated to stop transmitting the pulses and start reading the echoed ones, by using the function `PulseIn()`, where it measures the time between changing the state for the echo pin from low to high. And this time represents the time to transmit and receive

pulses from the object, so in order to find the distance it must be divided by 2, and multiply it by 343m/s (0.0343cm/us) which is the speed of the pulses, where the result will be in cm.

```

32 // This function returns the distance for the object from the ultra
33 long getDistance(){
34     digitalWrite(TrigPin, LOW); //Deactivate the Trigger pin.
35     delayMicroseconds(2); // Wait for 2 microseconds
36     digitalWrite(TrigPin, HIGH); //Activate the Trigger pin, to send
37     delayMicroseconds(10); // Wait for 10 microseconds
38     digitalWrite(TrigPin, LOW); //Deactivate the Trigger pin to read
39     // pulseIn measures the time since the EchoPin changes its status
40     // it returns the sound wave travel time in microseconds
41     time = pulseIn(EchoPin, HIGH);
42     //Speed = 0.0343/2 = 0.01715;
43     return 0.01715 * time;
44 }
45

```

Serial Monitor

11:18 PM
5/21/2021

Figure 13: Get Distance function Code

To open the door, a 90° rotation for the servo motor was chosen to represent it, and to let it open completely without interruption, a 1s delay will execute, followed by a 3s delay to keep the door open during it. While, in closing the door, the servo motor will rotate to 0, which is the original state, followed by 1s delay to insure the rotation. And in the loop function, it will only close the door when the distance >50, where it will enable to check every time if the door should remain open or be closed. The doorOpen variable is to make sure the door is closed and will be used later.

```

46 //This function to open the door wich is the servo motor
47 void openDoor(){
48     servo.write(90); //Rotates the servo motor to 90 degree.
49     delay(1000); //to let the servo motor rotate before making any fu
50     delay(3000); //let the servo motor open for 3 sec
51 }
52 //This function to close the door wich is the servo motor
53 void closeDoor(){
54     servo.write(0); //Rotates the servo motor to the original (0 degr
55     delay(1000); //to let the servo motor rotate before making any fu
56     doorOpen=0;
57 }
58

```

Serial Monitor

11:48 PM
5/21/2021

Figure 14: Door Functions Code

LDR, PIR, and the Bulb:

For the implementation, the LDR first terminal is connected to analog pin and the GND via a 10kΩ resistor, and the other pin is connected to the 5v. The PIR is connected to the GND and 5v and a digital pin. Finally, the bulb is connected to a digital pin by one pin and to the ground by the other pin.

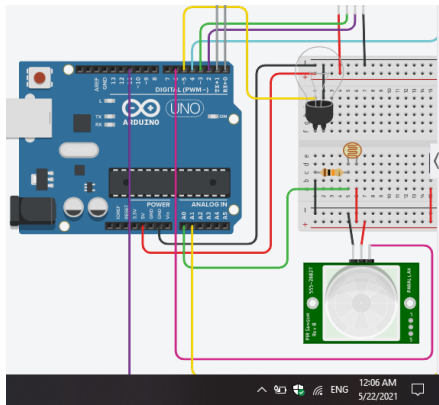


Figure 15: LDR, PIR, and the Bulb Circuit

To get the values of the devices, the two functions in the figure were used.

```

77  LDRReading = analogRead(LDRPin); //Read the value from the LDR s
78  PirReading = digitalRead(PirPin); //Read the value from the PIR
79

```

Serial Monitor

12:14 AM
5/22/2021

Figure 16: get LDR and PIR values Code

When there is a movement (PIR in high state) and there is no enough light ($LDR < 600$), then the light bulb must be turned on for 1 minute using the function `digitalWrite (bulbPin, HIGH)`. If the delay function was used to turn on the light for 1 min, it would have decreased the efficiency of the project, because 1min is a good amount of time, where many actions could occur and they need urgent handling. Therefore, the function `millis ()` is used, where it returns the number of milliseconds passed since the Arduino board began running the current program, and by initializing it when turning on the light then writing an if statement and comparing the difference between the starting milliseconds and the current one with 60000(1min/60s), it will allow for the bulb to be on for approximately 1min. Then the bulb will either turn off, or continue on for another 1min depending on the state of the PIR and LDR.

```

94  if (LDRReading < 600 && PirReading == HIGH){
95      digitalWrite(bulbPin, HIGH); //To turn on the bulb
96      //millis() return number of milliseconds passed since the Ardui
97      //it is used because waiting for 1 minute using the delay witho
98      startMillis = millis();
99  }
100 //check if the bulb is ON
101 //then check if 1 minute has been spent since turning on the bulb
102 if(digitalRead(bulbPin) == HIGH && millis()-startMillis >=60000){
103     //if the minute finished but there is still movement & there is n
104     //the bulb will be ON for another 1 min
105     if(LDRReading < 600 && PirReading == HIGH)
106         startMillis = millis();
107     //if there is no movement or there is enough light, the bulb wi
108     else{
109         digitalWrite(bulbPin, LOW); //To turn off the bulb
110         startMillis=0;
111     }
112

```

Serial Monitor

12:18 AM
5/22/2021

Figure 17: LDR and PIR Code

Temperature Sensor and DC Motor:

The temperature sensor (Tnp36) is connected to the 5v, GND, and an analog pin (digital pin #11). While the DC motor is connected to the circuit using a resistor, diode, NPN transistor, and a voltage source. Which is considered a driving circuit for the motor due to its high current, where the BJT transistor is a driver for the DC motor, the resistor on the base to limit the current from the digital pin, and the diode is needed to protect the transistor.[1]

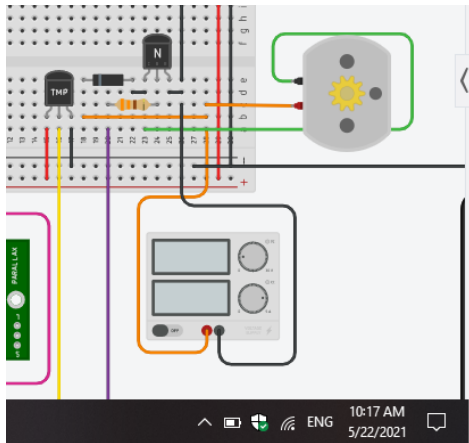


Figure 18: Tmp36 and DC motor Circuit

To get the temperature value in Celsius, the analog input which ranges from 0-1023 will be mapped to the range of the sensor which is -40 -125.

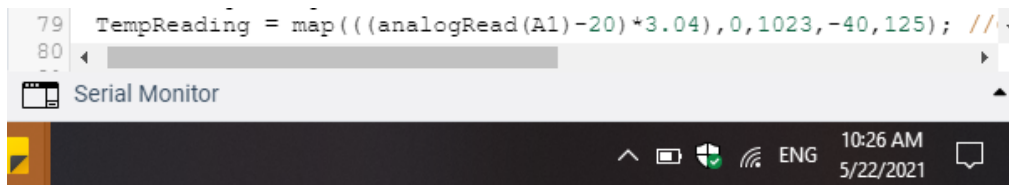


Figure 19: Obtain Temperature value Code

When the temperature is above 30, the DC motor will be turned on with a speed depending on the temperature and it is chosen to be twice the temperature.

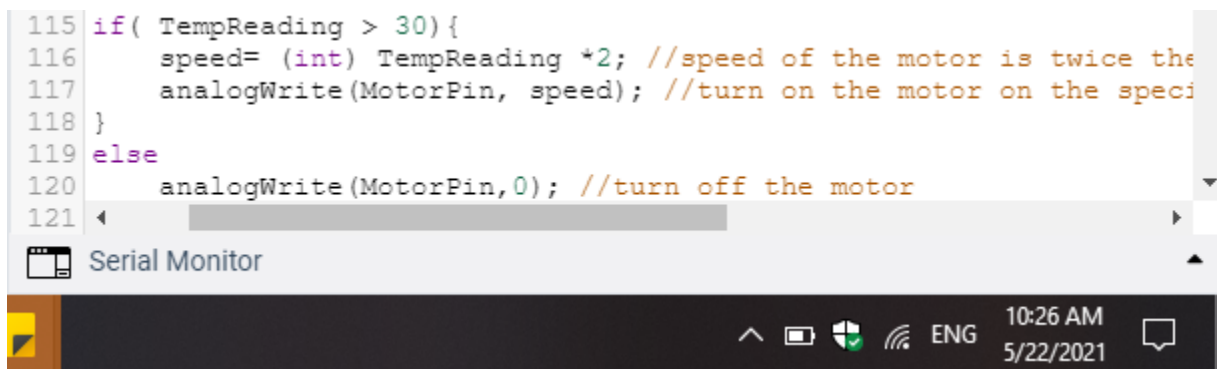


Figure 20: Tmp36 Code

The Switch:

The switch is connected to the 5v, GND, and a digital pin, the right side is high state and it represents the auto mode while the left side with low state represents the manual mode.

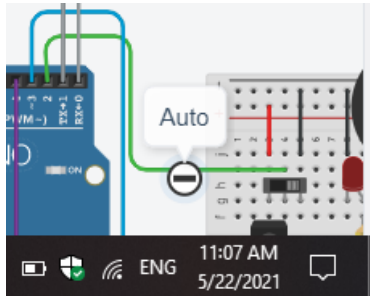


Figure 21: Switch Circuit

The code for the switch which is in the second Arduino code, and it consists of reading the state of the switch, where if it is high then the value is -1 and the mode is auto, while if it is low then the value is 1 and the mode is manual. And to not keep sending the switch value to the other Arduino and only send it once, a previousValue variable is initialized which will have the value of the switch, where if the new switch value is the same as the previousValue, the value will not be send to the other switch.

```
31 void loop()
32 {
33     switchValue = digitalRead(switchPin); //get the value of the sw
34     //if the switch is high it means it is in auto mode
35     //also check if -1 was sent before to not send it again
36     //so both arduinos work fast
37     if (switchValue == HIGH && previousValue!=-1) {
38         Serial.println(-1); //auto
39         previousValue=-1;
40     }
41     //if the switch is low it means it is in manual mode
42     else if (switchValue == LOW && previousValue!=1){
43         Serial.println(1); //manual
44         previousValue=1;
45     }
46 }
```

Serial Monitor

ENG 11:25 AM 5/22/2021

Figure 22: Switch Code

LEDs and Buzzers:

Three LEDs and Buzzers were used to notify the user to do actions if needed, where the red LED and the buzzer next to it is for the ultrasonic sensor, the yellow LED and the buzzer besides it for the PIR and LDR, and the white LED and the buzzer next to it is for the temperature sensor.

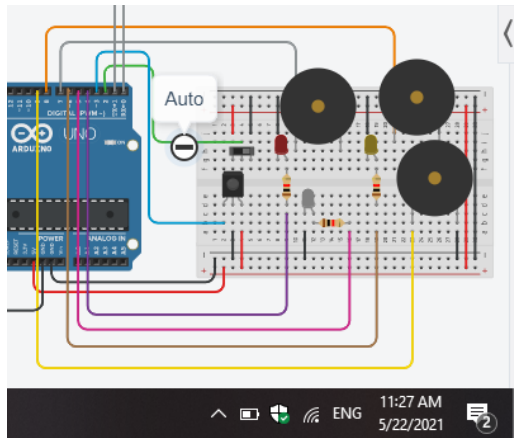


Figure 23: LEDs and Buzzers Circuit

When the mode is manual, the first Arduino will send signals if any action is needed, where it will send 2 if the servo motor needs to be open, and then the LED and buzzer will be turned on to notify the user to press the button on the remoter control. And signal 3 is to turn on the bulb while signal 4 to turn on the DC motor. For the buzzer, the function Tone could be used to specify a specific tone, and it would be more efficient to create different tones for different signal. But the tone function conflicts with the IRRemote, where both of them use the same timer, and it results in an error, and if the project was not implemented in TinkerCad, this could be solved by changing the timer in the IRRemote.h library.

```

46 //the manual mode (controlling the remote/leds/buzzers)
47 if (switchValue == LOW) {
48   if (Serial.available()) { //if the other arduino sent an signal
49     int data= Serial.parseInt(); //save it as integer
50     //signal 2 for the servo motor (door)
51     //it means that the distance >=50 and the user must be noti
52     if(data==2){
53       digitalWrite(servoLEDPin, HIGH); //To turn on the led for
54       digitalWrite(servoBuzPin,HIGH); //To turn on the buzzer f
55     }
56     //signal 3 for the bulb
57     //it means that there was movement and no enogh light and t
58     if(data==3){
59       digitalWrite(bulbLEDPin, HIGH); //To turn on the led for
60       digitalWrite(bulbBuzPin,HIGH); //To turn on the buzzer fc
61     }
62     //signal 4 for the motor
63     //it means that the temperature >30 and the user must be nc
64     if(data==4){
65       digitalWrite(motorLEDPin, HIGH); //To turn on the led for
66       digitalWrite(motorBuzPin,HIGH); //To turn on the buzzer f
67     }
68   }
69 }

```

Serial Monitor

Figure 24: LEDs and Buzzers Code

Remote Control:

To enable the remote control, an IR sensor must be used to connect the remote with the Arduino. The function `irrecv.decode()` is used which returns true if anything is received, and stores it in variable results. Then using the value of the result, it determines which button was pushed as shown in the figure, and sends a specific signal to the first Arduino to do the action. Where

button 1 (value: 2295) is pushed to open the door, button 2 (value:34935) is used to turn on the bulb, and button 3 (value: 18615) is used to turn on the DC motor.

```

69 //for the IR Remote
70 if (irrecv.decode(&results)) { //returns true if anything is recieved
71   unsigned int value = results.value; //Get the value of results
72   switch (value) {
73     case 2295: // button #1
74       digitalWrite(servoLEDPin, LOW); //turn off the servo led
75       digitalWrite(servoBuzPin, LOW); //turn off the servo buzzer
76       Serial.println(5); //send the other arduino signal 5 to open the door
77       break; //so the other cases don't run
78     case 34935: //button #2
79       digitalWrite(bulbLEDPin, LOW); //turn off the bulb led
80       digitalWrite(bulbBuzPin, LOW); //turn off the bulb buzzer
81       Serial.println(6); //send the other arduino signal 6 to turn on the bulb
82       break;
83     case 18615: //button #3
84       digitalWrite(motorLEDPin, LOW); //turn off the motor led
85       digitalWrite(motorBuzPin, LOW); //turn off the motor buzzer
86       Serial.println(7); //send the other arduino signal 7 to turn on the motor
87       break;
88   }
89   irrecv.resume(); // Receive the next value
90 }
91
92
93

```

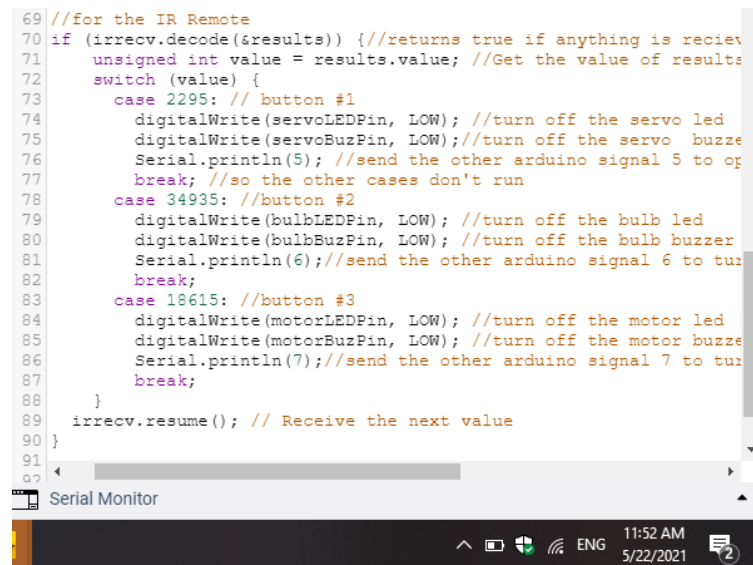


Figure 25: IR Remote and IR Sensor Code

Manual Mode:

When the second Arduino sends the signals about which mode to work in, the first Arduino reads the signal from the serial, and if it was -1 then the mode is auto while 1 is for manual mode as mentioned before. And because the second Arduino sends the signal about the mode only once for speed efficiency, the first Arduino read the signal about the mode and save it in a variable called previousData, so when there is no signal (data=0), it will be able to execute the functions for the specific mode as shown in the figure below.

```

86   if(data== -1 || (data==0 && previousData== -1)){
87     if ( distance <= 50) //if the distance is less or equal
88

```

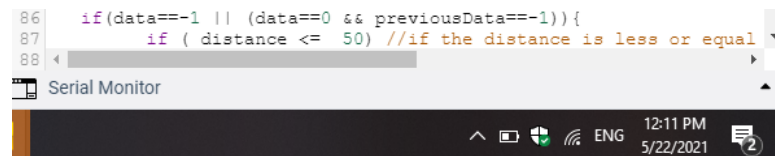


Figure 26: Manual Code 1

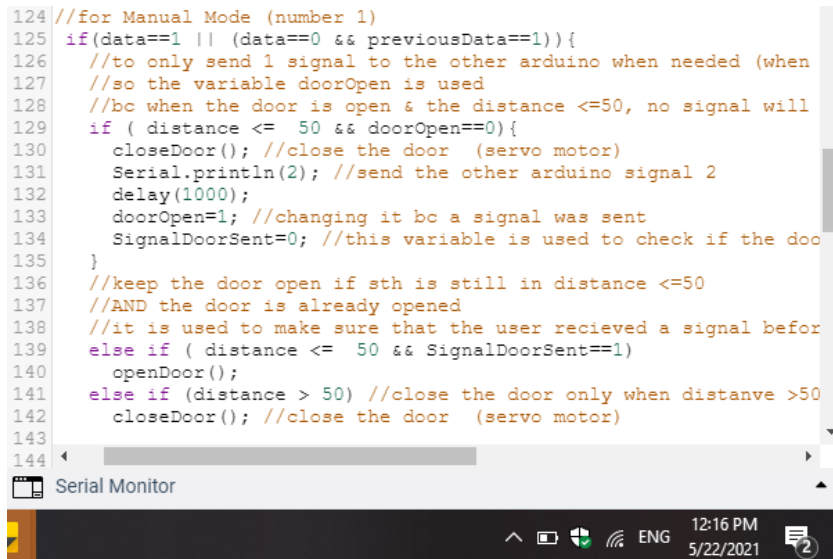
In the manual mode, when the user needs to be notified about an action to do, a signal will be sent from the first Arduino to the second, and to only send the signal once for 1 action, for example when there is an object in distance less than 50, a signal must be used to the second Arduino to open the door once, where even if the object still exist, the user has to do nothing, and the door will keep open for 3s each time automatically. And also, when the temperature rises while the DC motor is running, the user doesn't have to be notified and the speed of the motor should increase automatically. Therefore, variables were initialized to help execute the latter, where in the case of the door and ultrasonic sensor, first, the variable doorOpen was initialized to show that the signal is sent and the door will be opened, and to not send the signal again, and it be set to the initial value after the door is closed, which means that now if there is an object in distance ≤ 50 , the user will be notified because the first object was moved. Also, the signal

SignalDoorSent is used to allow keep the door open after 3s if the object is still in distance less than 50cm, and it will be reset when the signal is sent and set after the door opens. Finally, to make the door close only if the object at distance >50, the if statement was used.

```

124 //for Manual Mode (number 1)
125 if(data==1 || (data==0 && previousData==1)){
126     //to only send 1 signal to the other arduino when needed (when
127     //so the variable doorOpen is used
128     //bc when the door is open & the distance <=50, no signal will
129     if ( distance <= 50 && doorOpen==0){
130         closeDoor(); //close the door (servo motor)
131         Serial.println(2); //send the other arduino signal 2
132         delay(1000);
133         doorOpen=1; //changing it bc a signal was sent
134         SignalDoorSent=0; //this variable is used to check if the doo
135     }
136     //keep the door open if sth is still in distance <=50
137     //AND the door is already opened
138     //it is used to make sure that the user recieved a signal befor
139     else if ( distance <= 50 && SignalDoorSent==1)
140         openDoor();
141     else if (distance > 50) //close the door only when distanve >50
142         closeDoor(); //close the door (servo motor)
143
144

```



The screenshot shows an IDE window with a code editor containing lines 124 to 144 of code. Below the code editor is a Serial Monitor window. The system tray at the bottom shows the time as 12:16 PM on 5/22/2021.

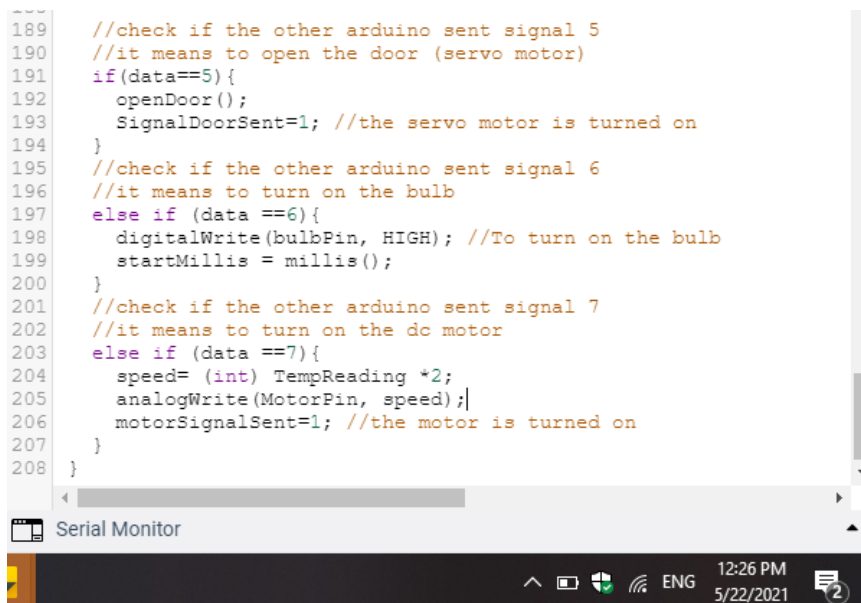
Figure 27: Manual Code 2

When the user sends the signals to do the actions, the following figure shows the code responsible to do them, where they are the first step in the actions, and the next steps which are to recheck the sensors to determine the next steps are located in the part when the user sends 1 for manual mode or sends nothing (data=0) but the mode is manual.

```

189 //check if the other arduino sent signal 5
190 //it means to open the door (servo motor)
191 if(data==5){
192     openDoor();
193     SignalDoorSent=1; //the servo motor is turned on
194 }
195 //check if the other arduino sent signal 6
196 //it means to turn on the bulb
197 else if (data ==6){
198     digitalWrite(bulbPin, HIGH); //To turn on the bulb
199     startMillis = millis();
200 }
201 //check if the other arduino sent signal 7
202 //it means to turn on the dc motor
203 else if (data ==7){
204     speed= (int) TempReading *2;
205     analogWrite(MotorPin, speed);|
206     motorSignalSent=1; //the motor is turned on
207 }
208 }

```



The screenshot shows an IDE window with a code editor containing lines 189 to 208 of code. Below the code editor is a Serial Monitor window. The system tray at the bottom shows the time as 12:26 PM on 5/22/2021.

Figure 28: Manual Code 3

Testing

1) Automatic Mode:

In automatic mode, Arduino 2 only sends signal -1 to Arduino 1, to make it work on automatic mode, and Arduino 1 didn't send any signal.

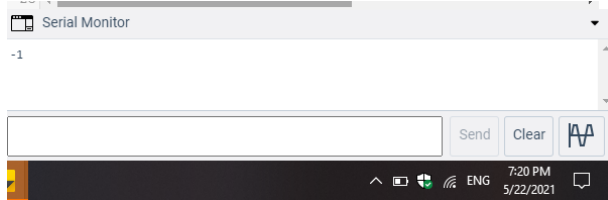


Figure 29: Auto Mode Test- Serial Monitor 1

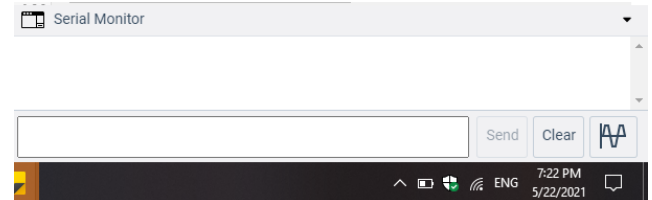


Figure 30: Auto Mode Test- Serial Monitor 2

The temperature sensor detected a temperature above 30°C , so the dc motor was turned on a speed equals to twice the temperature as shown in figure, then an object was detected by the ultrasonic sensor at distance 39.6cm which is less than 50cm, so the servo motor (door) opened (angle 90°) as shown in figure. Also, at time 11 in TinkerCad timer, the LDR detected that there was no light ($\text{light} < 600$) and there was a movement observed by the PIR, thus the bulb was turned on.

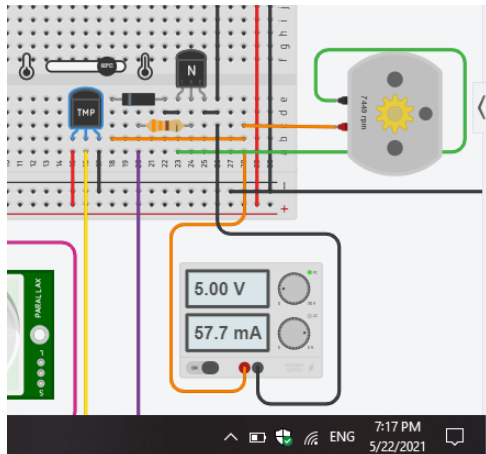


Figure 31: Auto Mode Test- Fan Test

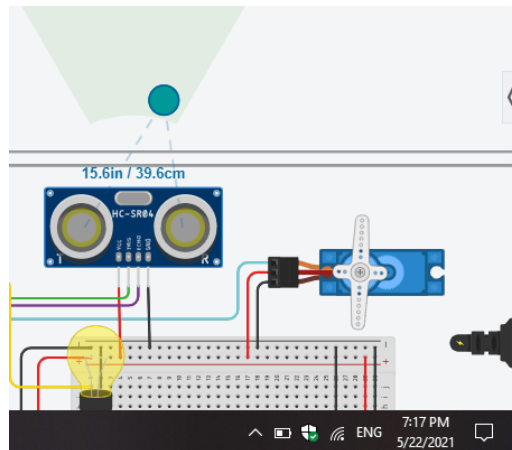


Figure 32: Auto Mode Test- Door Test

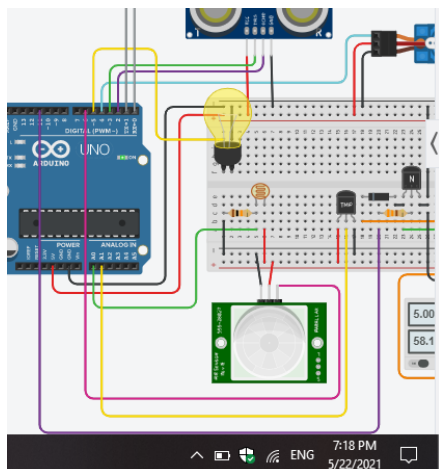


Figure 33: Auto Mode Test- Bulb Test

The figure below shows all the results together:

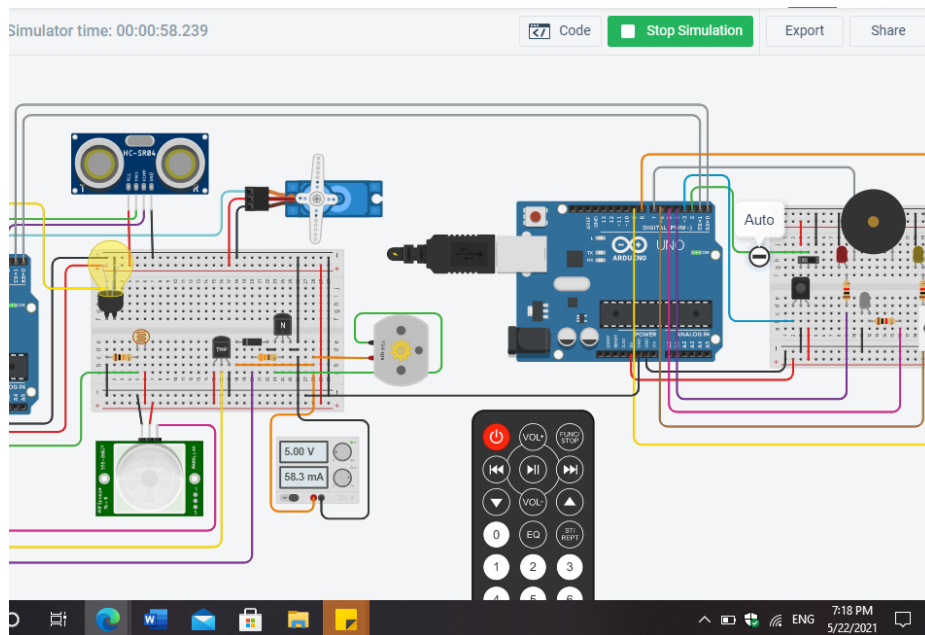


Figure 34: Auto Mode Test- All results 1

After the latter, the object in the ultrasonic sensor was moved to distance >50 , hence the door closed, the temperature was lowered, which decreased the speed of the dc motor (it can be observed clearly by the decrease of the current for the power supply), and after 1min (timer 1:11), the bulb was turned off, after checking that there was no movement.

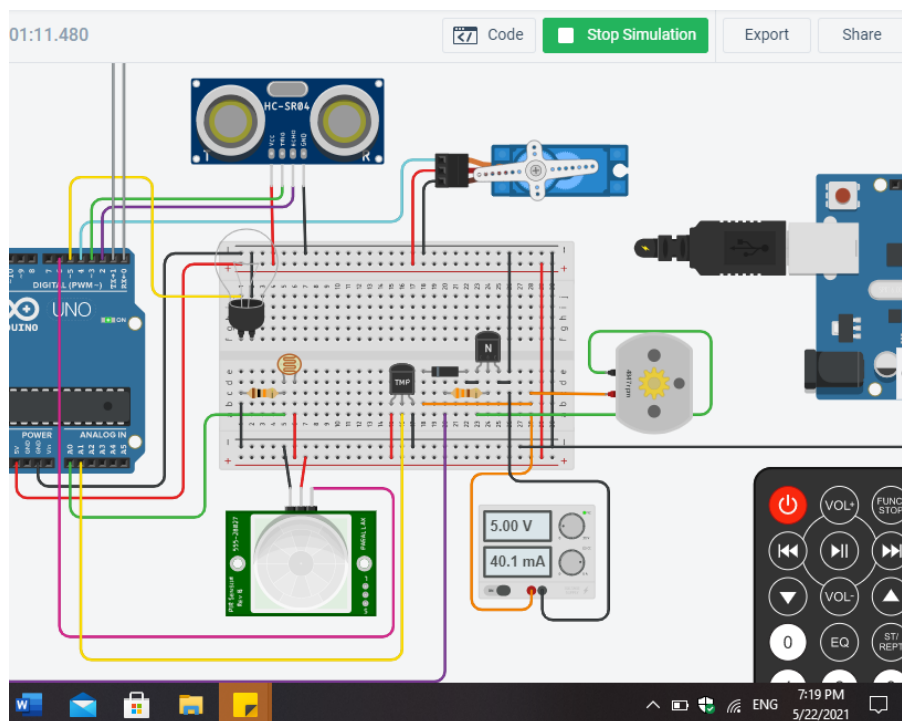


Figure 35: Auto Mode Test- All results 2

2) Manual Mode:

First, the switch was set to low which is the manual mode, thus Arduino 2 sent Arduino 1 signal 1, then the object in ultrasonic sensor was set to distance 40.4cm which is less than 50cm, and the door needs to be open, so Arduino 1 sent signal (2) to Arduino 2, to notify the user, hence the first buzzer and the red LED were turned on, then when the user pressed button 1 on the remote control, and Arduino 2 send to Arduino 1 the signal 5 to open the door, and then the servo motor was turned to 90°; to open the door.

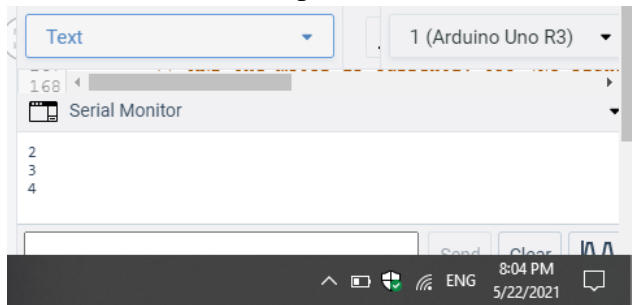


Figure 36: Manual Mode Test- Serial Monitor 1

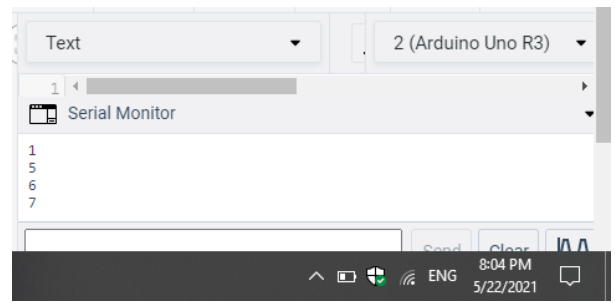


Figure 37: Manual Mode Test- Serial Monitor 2

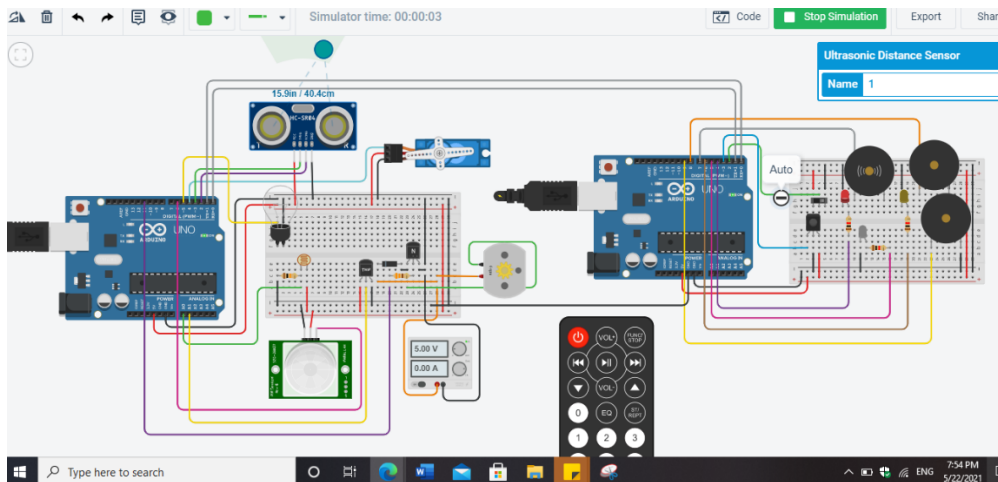


Figure 38: Manual Mode Test- Door Test

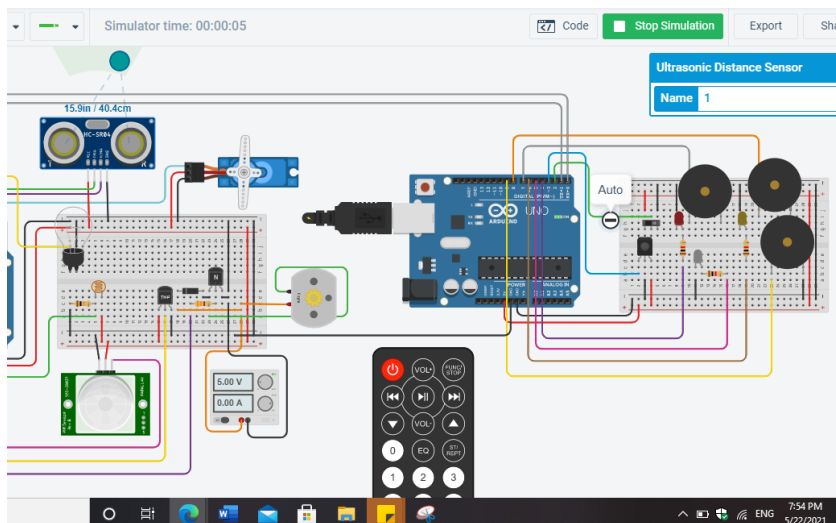


Figure 39: Manual Mode Test- Door Result

Next, the object in the PIR was moved, and the light in the LDR was set to <600 (low), thus the bulb needs to be turned on, so Arduino 1 sent signal (3) to Arduino 2, to notify the user, hence the second buzzer and the yellow LED were turned on, then when the user pressed button 2 on the remote control, the buzzer and the yellow LED were turned off and Arduino 2 sent to Arduino 1 the signal 6 to turn on the bulb, then the bulb was turned on (at simulator time 0:22) for 1min, and it turned off at simulator time 1:22, as shown in the figure.

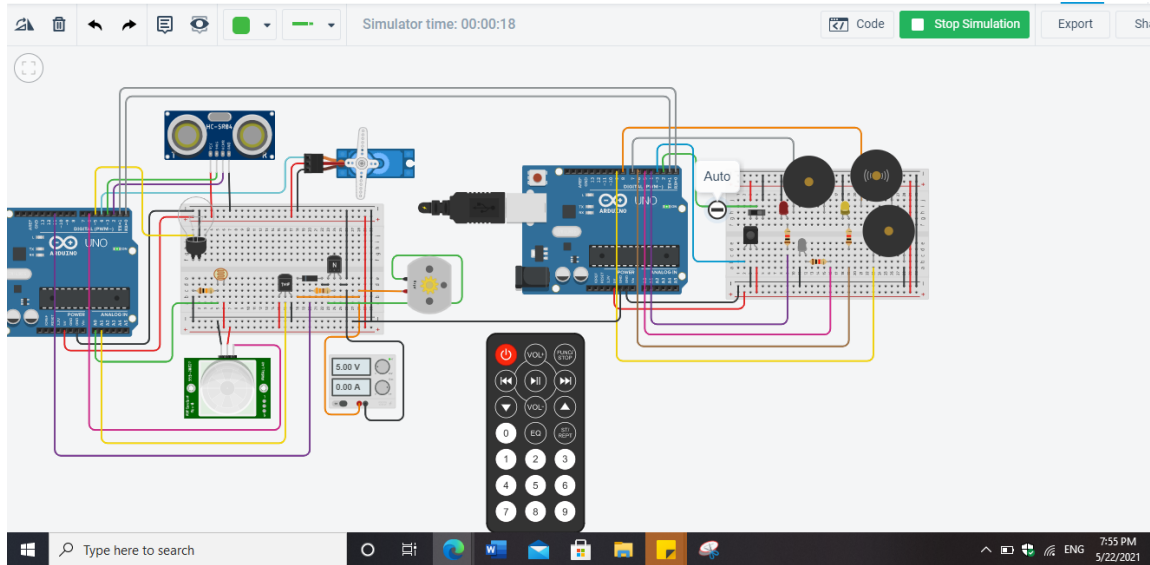


Figure 40: Manual Mode Test- Bulb Test

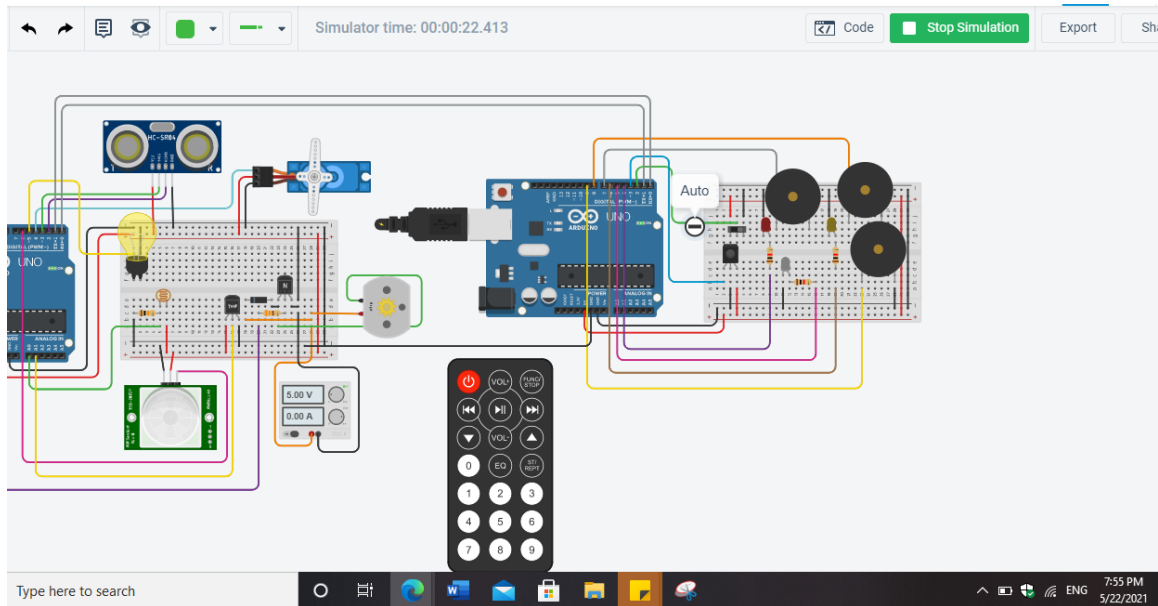


Figure 41: Manual Mode Test- Bulb Result 1

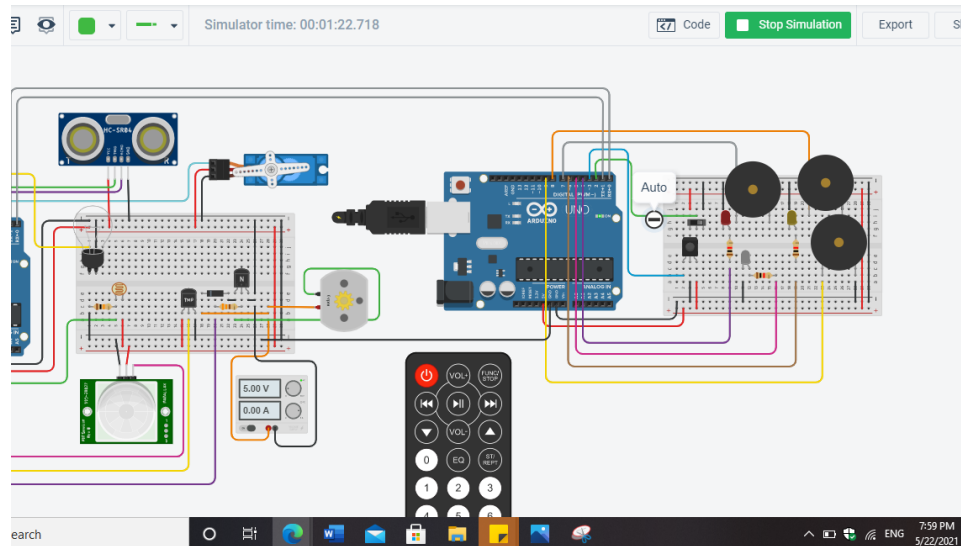


Figure 42: Manual Mode Test- Bulb Result 2

Later on, the temperature increased above 30°C , which requires to turn on the fan (servo motor), so Arduino 1 sent signal (4) to Arduino 2, to notify the user, hence the third buzzer and the white LED were turned on, then when the user pressed button 3 on the remote control, the buzzer and the white LED were turned off and Arduino 2 sent to Arduino 1 the signal 7 to turn on the fan.

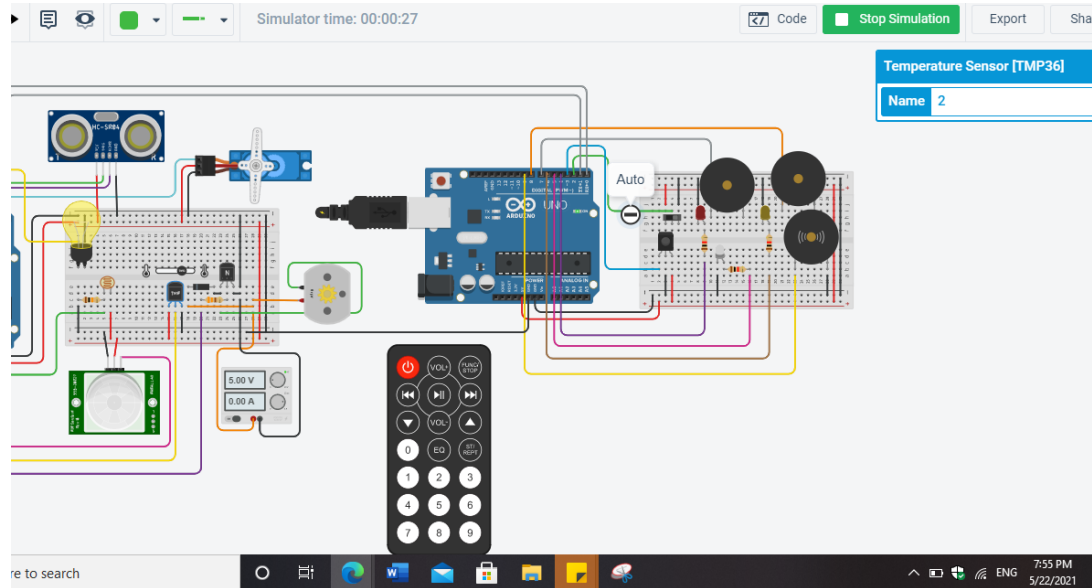


Figure 43: Manual Mode Test- DC Motor Test

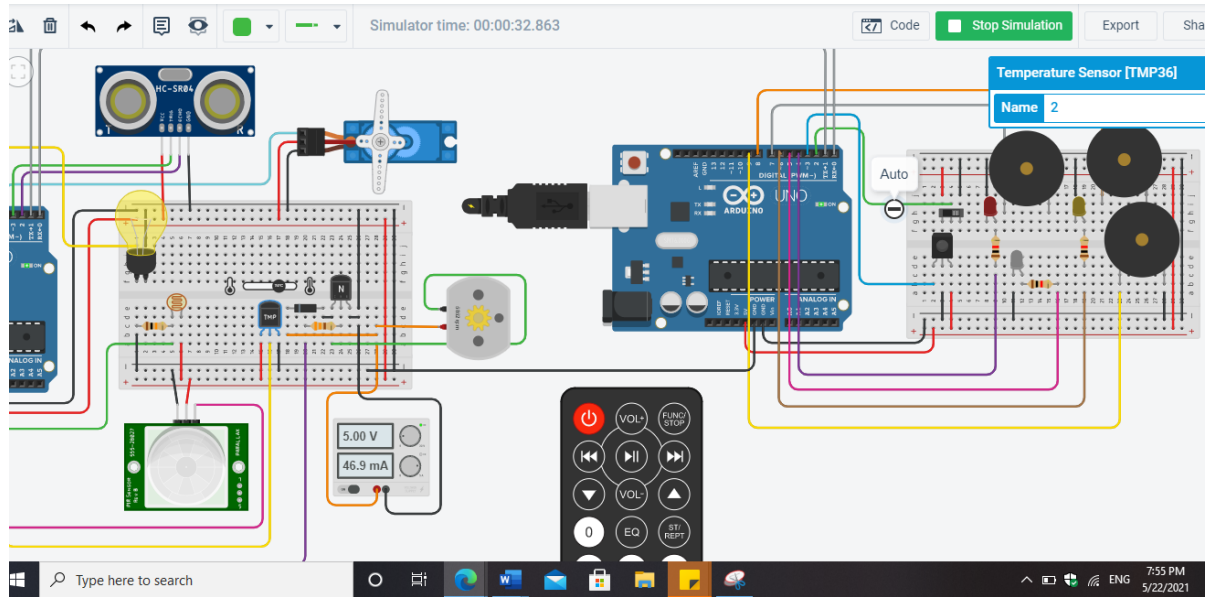


Figure 44: Manual Mode Test- DC Motor Result

Finally, after the temperature decreased to lower than 30°C, and the object moved to distance above 50cm, the door was closed and the fan was turned off.

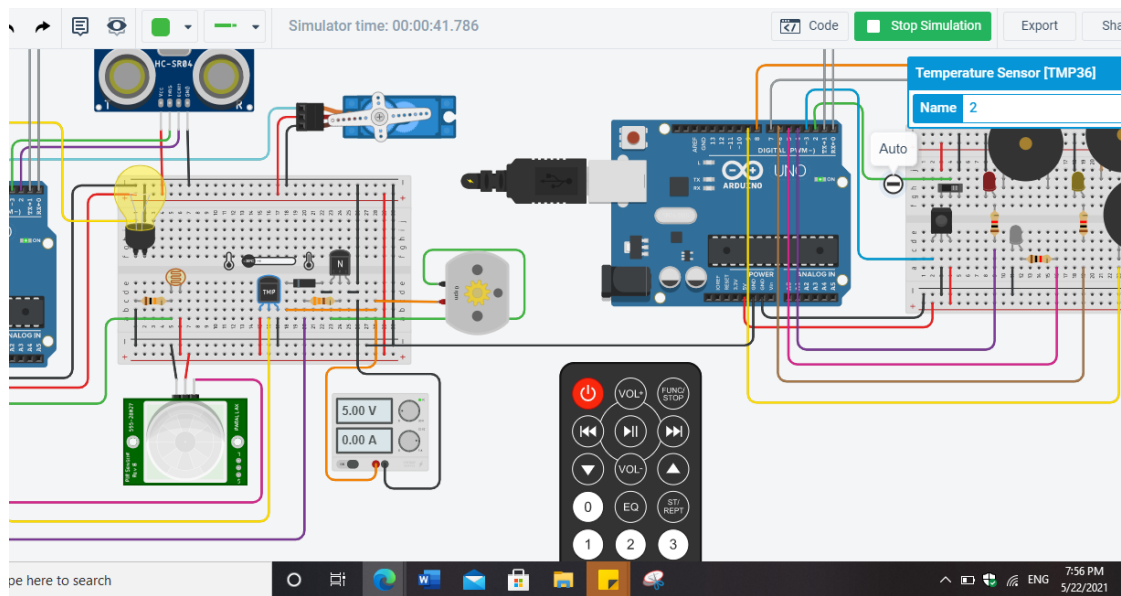


Figure 45: Manual Mode Test- Final Result

Tinkercad Project Link:

https://www.tinkercad.com/things/gb1g61fEDwB-project/editel?sharecode=nz36AXFia4QQ-m_Qj40ZaEgcAgU-HbrmCGBuzOaV2kY

Conclusion

From this project, it was concluded how to use the Arduino to build a compatible project, and how to synchronize between two Arduinos using the serial communication. Also, how to use many sensors such as the PIR, LDR, temperature sensor, and ultrasonic sensor, to connect them to the Arduino, and to obtain the values from them. Additionally, how to rotate the servo motor on different angles, and how to rotate the DC motor on different speeds. Moreover, how to use the IR remote, and IR sensor, and obtain the code for each button. Furthermore, that millis function could be used instead of the delay function if needed, such when a blockage of the code affects the performance when the wait time is large using the delay function. Finally, that some devices require to be connected to the analog pins such as the LDR and the DC motor.

References

- [1] Computer Design Lab Manual.
- [2] <https://learn.sparkfun.com/tutorials/what-is-an-arduino/all>
Accessed on 21-5-2021 at 7:30 pm.
- [3] <https://www.instructables.com/Ultrasonic-Distance-Sensor-Arduino-Tinkercad/>
Accessed on 21-5-2021 at 7:45 pm.
- [4] <https://create.arduino.cc/projecthub/ioarvanit/measuring-distance-with-sound-353c17>
Accessed on 22-5-2021 at 8:30 pm.
- [5] <https://www.instructables.com/Arduino-Servo-Motors/>
Accessed on 22-5-2021 at 8:45 pm.
- [6] <https://maker.pro/arduino/tutorial/how-to-use-an-ldr-sensor-with-arduino#:~:text=The%20LDR%20is%20a%20special,DIY%20Arduino%20LDR%20sensor%20project.>
Accessed on 22-5-2021 at 9:10 pm.
- [7] <https://www.instructables.com/PIR-Motion-Sensor-With-Arduino-in-Tinkercad/>
Accessed on 22-5-2021 at 9:25 pm.
- [8] <https://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor>
Accessed on 22-5-2021 at 9:45 pm.
- [9] [How To Control a DC Motor with an Arduino - Projects \(allaboutcircuits.com\)](http://allaboutcircuits.com/projects/how-to-control-a-dc-motor-with-an-arduino/)
Accessed on 22-5-2021 at 10:10 pm.
- [10] <https://www.makerguides.com/ir-receiver-remote-arduino-tutorial/>
Accessed on 22-5-2021 at 10:30 pm.

Appendix

Code for the First Arduino:

```
//The library for physical motors
#include <Servo.h>

//Define Pins:
int TrigPin = 3; //For the Trigger pin in the ultrasonic sensor
int EchoPin = 2; //For the Echo pin in the ultrasonic sensor
int ServoPin = 4; //for the Servo motor
int bulbPin = 5; //for the bulb
int LDRPin = A0; // For the LDR (photoresistor) which is connected to a0
int PirPin = 6; //for the PIR Sensor
int TempPin = A1; //for the temperature sensor
int MotorPin = 11; //for the DC motor

//Define variables:
long time;
long distance;
Servo servo; //Assigns the Servo type variable to a physical motor with name servo
int LDRReading; // the analog reading from the photoresistor
int PirReading;
float TempReading;
int data=9;
int previousData=9;
int speed;
int doorOpen=0;
int SignalDoorSent=0;
int motorON=0;
int motorSignalSent=0;
int bulbON=0;
unsigned long startMillis = 0;

// This fuction returns the distance for the object from the ultrasonic sensor.
long getDistance(){
    digitalWrite(TrigPin, LOW); //Deactivate the Trigger pin.
    delayMicroseconds(2); // Wait for 2 microseconds
    digitalWrite(TrigPin, HIGH); //Activate the Trigger pin, to send the pulse
    delayMicroseconds(10); // Wait for 10 microseconds
    digitalWrite(TrigPin, LOW); //Deactivate the Trigger pin to read the pulse.
    // pulseIn measures the time since the EchoPin changes its status from low to high
```

```

// it returns the sound wave travel time in microseconds
time = pulseIn(EchoPin, HIGH);
//Speed = 0.0343/2 = 0.01715;
return 0.01715 * time;
}

//This function to open the door wich is the servo motor
void openDoor(){
  servo.write(90); //Rotates the servo motor to 90 degree.
  delay(1000); //to let the servo motor rotate before making any further moves
  delay(3000); //let the servo motor open for 3 sec
}

//This function to close the door wich is the servo motor
void closeDoor(){
  servo.write(0); //Rotates the servo motor to the original (0 degree)
  delay(1000); //to let the servo motor rotate before making any further moves
  doorOpen=0;
}

void setup(){
  Serial.begin(9600); //For communication with the other arduino
  pinMode(TrigPin, OUTPUT); //define the Trigger pin as output
  pinMode(EchoPin, INPUT); //define the Echo Pin as input
  servo.attach(ServoPin); //attach the servo motor to the pin
  pinMode(LDRPin, INPUT); //define A0 as input which will be connected to the LDR
  pinMode(PirPin, INPUT); //Define PIR pin as input
  pinMode(bulbPin, OUTPUT); //Define the Bulb pin as output
  pinMode(TempPin, INPUT); //Define Temperature sensor pin as input
  pinMode(MotorPin, OUTPUT); //Define the Motor pin as output
}

void loop(){
  //check if the arduino sent data
  if(Serial.available())
    data = Serial.parseInt(); // save the data as int type

  distance = getDistance(); //get the distance in the ultrasonic sensor
  LDRReading = analogRead(LDRPin); //Read the value from the LDR sensor
  PirReading = digitalRead(PirPin); //Read the value from the PIR sensor
  TempReading = map(((analogRead(A1)-20)*3.04),0,1023,-40,125); //Get the reading from the
  temperature sensor

  //for automatic Mode (number -1)

```

```

//data=0, when there is no data sent from the other arduino
//which means no changes has been made
//then what defines the mode is the variable previousData
//if it equals -1 > auto mode
if(data== -1 || (data==0 && previousData== -1)){
    if ( distance <= 50) //if the distance is less or equal 50cm
        openDoor(); //open the door (servo motor)
    else // else (if the distance larger than 50cm)
        closeDoor(); //close the door (servo motor)

    // Check if the there is no enough light
    //AND there is a movement (PIR high)
    if (LDRReading <600 && PirReading == HIGH){
        digitalWrite(bulbPin, HIGH); //To turn on the bulb
        //millis() return number of milliseconds passed since the Arduino board
        began running the current program
        //it is used because waiting for 1 minute using the delay without doing anything else is
        inefficient
        startMillis = millis();
    }
    //check if the bulb is ON
    //then check if 1 minute has been spent since turning on the bulb
    if(digitalRead(bulbPin) == HIGH && millis()-startMillis >=60000){
        //if the minute finished but there is still movement & there is no enough light
        //the bulb will be ON for another 1 min
        if(LDRReading <600 && PirReading == HIGH)
            startMillis = millis();
        //if there is no movement or there is enough light, the bulb will be turned off
        else{
            digitalWrite(bulbPin, LOW); //To turn off the bulb
            startMillis=0;
        }
    }

    //check the temperature if it is above 30 to turn on the motor
    if( TempReading > 30){
        speed= (int) TempReading *2; //speed of the motor is twice the temperature to
        make relation between speed and temp
        analogWrite(MotorPin, speed); //turn on the motor on the specific speed
    }
    else
        analogWrite(MotorPin,0); //turn off the motor

```

```

    previousData=-1;
}

//for Manual Mode (number 1)
if(data==1 || (data==0 && previousData==1)){
    //to only send 1 signal to the other arduino when needed (when distance <=50)
    //so the variable doorOpen is used
    //bc when the door is open & the distance <=50, no signal will be sent bc the user already
opened the door
    if ( distance <= 50 && doorOpen==0){
        closeDoor(); //close the door (servo motor)
        Serial.println(2); //send the other arduino signal 2
        delay(1000);
        doorOpen=1; //changing it bc a signal was sent
        SignalDoorSent=0; //this variable is used to check if the door is open or not
    }
    //keep the door open if sth is still in distance <=50
    //AND the door is already opened
    //it is used to make sure that the user recieved a signal before to open the door
    else if ( distance <= 50 && SignalDoorSent==1)
        openDoor();
    else if (distance > 50) //close the door only when distance >50
        closeDoor(); //close the door (servo motor)

    //to check if there is movement AND there is no enough light
    //AND no signal was sent before for this case (before turning the bulb off)
    if (LDRReading <600 && PirReading == HIGH && bulbON==0 ){
        digitalWrite(bulbPin, LOW); //To turn off the bulb
        Serial.println(3); //send the other arduino signal # 3
        delay(1000);
        bulbON=1; //this variable is used to check if the bulb is on or off
    }
    // to check when the bulb is on after 1min to either turn it off or keep it on for another 1min
    if(digitalRead(bulbPin) == HIGH && (millis()-startMillis >=60000)){
        //if the minute finished but there is still movement & there is no enough light
        //the bulb will continue ON for another 1 min
        if(LDRReading <600 && PirReading == HIGH)
            startMillis = millis();
        //if there is no movement or there is enough light, the bulb will be turned off
        else{
            digitalWrite(bulbPin, LOW); //To turn off the bulb
            startMillis=0; //set the start time to 0
        }
    }
}

```

```

    bulbON=0; //to allow to send another signal to the 2nd arduino if needed
  }
}

//Send signal only when temperature above 30
// AND the motor is currently OFF (No signal was sent for this case)
if( TempReading > 30 && motorON==0){
  analogWrite(MotorPin,0); //turn off the motor
  Serial.println(4); //send the other arduino signal #4
  delay(1000);
  motorON=1; //changing it bc a signal has been sent
  motorSignalSent=0; //this variable is used to check if the motor is on or off
}
// to see if the temperature has changed so the motor's speed will change
else if( TempReading > 30 && motorSignalSent==1){
  speed= (int) TempReading *2;
  analogWrite(MotorPin, speed);
}
//to turn off the motor when the temperature is less than 30
else if( TempReading <= 30){
  analogWrite(MotorPin,0);
  motorON=0;
}
previousData=1;
}

//check if the other arduino sent signal 5
//it means to open the door (servo motor)
if(data==5){
  openDoor();
  SignalDoorSent=1; //the servo motor is turned on
}
//check if the other arduino sent signal 6
//it means to turn on the bulb
else if (data ==6){
  digitalWrite(bulbPin, HIGH); //To turn on the bulb
  startMillis = millis();
}
//check if the other arduino sent signal 7
//it means to turn on the dc motor
else if (data ==7){
  speed= (int) TempReading *2;

```

```

    analogWrite(MotorPin, speed);
    motorSignalSent=1; //the motor is turned on
  }
}

```

Code for the Second Arduino:

```

#include <IRremote.h>

//Define Pins
int RemotePin = 3; //For the IR remote
int switchPin = 2; //For the slide switch
int servoLEDPin = 4; //For the servo LED alarm
int motorLEDPin = 5; //For the motor LED alarm
int bulbLEDPin = 6; //for the bulb LED alarm
int servoBuzPin = 7; //for the servo buzzer alarm
int motorBuzPin = 9; // For the motor buzzer alarm
int bulbBuzPin = 8; //for the bulb buzzer alarm

IRrecv irrecv(RemotePin); // create a receiver object of the IRrecv class
decode_results results; // create a results object of the decode_results class
int switchValue=HIGH;
int previousValue=2;

void setup()
{
  Serial.begin(9600);
  pinMode(switchPin, INPUT);
  irrecv.enableIRIn();
  pinMode(servoLEDPin, OUTPUT); //define the servo LED pin as output
  pinMode(motorLEDPin, OUTPUT); //define the motor LED pin as output
  pinMode(bulbLEDPin, OUTPUT); //define the bulb LED pin as output
  pinMode(servoBuzPin, OUTPUT); //define the servo buzzer pin as output
  pinMode(motorBuzPin, OUTPUT); //define the motor buzzer pin as output
  pinMode(bulbBuzPin, OUTPUT); //define the bulb buzzer pin as output
}

void loop()
{
  switchValue = digitalRead(switchPin); //get the value of the switch
  //if the switch is high it means it is in auto mode
  //also check if -1 was sent before to not send it again
  //so both arduinos work fast

```

```

if (switchValue == HIGH && previousValue!=-1) {
    Serial.println(-1); //auto
    previousValue=-1;
}
//if the switch is low it means it is in manual mode
else if (switchValue == LOW && previousValue!=1){
    Serial.println(1); //manual
    previousValue=1;
}
//the manual mode (controlling the remote/leds/buzzers)
if (switchValue == LOW) {
    if(Serial.available()){ //if the other arduino sent an signal
        int data= Serial.parseInt(); //save it as integer
        //signal 2 for the servo motor (door)
        //it means that the distance >=50 and the user must be notified
        if(data==2){
            digitalWrite(servoLEDPin, HIGH); //To turn on the led for the servo motor
            digitalWrite(servoBuzPin,HIGH); //To turn on the buzzer for the servo motor
        }
        //signal 3 for the bulb
        //it means that there was movement and no enogh light and the user must be notified
        if(data==3){
            digitalWrite(bulbLEDPin, HIGH); //To turn on the led for the bulb
            digitalWrite(bulbBuzPin,HIGH); //To turn on the buzzer for the bulb
        }
        //signal 4 for the motor
        //it means that the temperature >30 and the user must be notified
        if(data==4){
            digitalWrite(motorLEDPin, HIGH); //To turn on the led for the motor
            digitalWrite(motorBuzPin,HIGH); //To turn on the buzzer for the motor
        }
    }
}
//for the IR Remote
    if (irrecv.decode(&results)) { //returns true if anything is received, and stores info in
variable results
        unsigned int value = results.value; //Get the value of results as an unsigned int, so we can
use switch case
        switch (value) {
            case 2295: // button #1
                digitalWrite(servoLEDPin, LOW); //turn off the servo led

```

```

    digitalWrite(servoBuzPin, LOW); //turn off the servo buzzer
    Serial.println(5); //send the other arduino signal 5 to open the door
    break; //so the other cases don't run
case 34935: //button #2
    digitalWrite(bulbLEDPin, LOW); //turn off the bulb led
    digitalWrite(bulbBuzPin, LOW); //turn off the bulb buzzer
    Serial.println(6); //send the other arduino signal 6 to turn on the bulb
    break;
case 18615: //button #3
    digitalWrite(motorLEDPin, LOW); //turn off the motor led
    digitalWrite(motorBuzPin, LOW); //turn off the motor buzzer
    Serial.println(7); //send the other arduino signal 7 to turn on the motor
    break;
}
irrcv.resume(); // Receive the next value
}
}
}

```