

Jacobs University
CH-220-B Introduction to Robotics
Fangning Hu
Lab Session 1
19 April 2021
Campus Ring 1, 28759 Bremen
Group 3, Cohort B -Simulation
Bazl-e- Fatima
Blen Assefa
Era Gërbeshi

Table of Contents

<i>1.1 Abstract</i>	<i>3</i>
<i>1.2 Theory</i>	<i>3</i>
<i>1.2.1 Arduino platform</i>	<i>3</i>
<i>1.2.2 Arduino software</i>	<i>3</i>
<i>1.3 Equipment</i>	<i>4</i>
<i>1.4 Procedure and execution</i>	<i>4</i>
<i>Task 1.1</i>	<i>4</i>
<i>Task 1.2</i>	<i>5</i>
<i>Task 1.3</i>	<i>1</i>
<i>Task 1.4</i>	<i>2</i>
<i>Task 1.6</i>	<i>2</i>
<i>Task 1.8</i>	<i>4</i>
<i>1.5 Discussion/Conclusion</i>	<i>1</i>
<i>References</i>	<i>1</i>

1.1 Abstract

This experiment deals with the investigation of the Arduino platform and additional measuring instruments. Alongside of that, it introduces the processing language and its applications, through simple circuit building. In this experiment the main goal is to get acquainted with the Arduino platform and programming the Arduino, with processing language.

1.2 Theory

1.2.1 Arduino platform

Arduino is a Physical or Embedded Computing platform based on easy-to-use hardware and software. It is also considered a tiny computer that can be programmed to process inputs and outputs between the device and external components that it is connected to.

The Arduino project was started at the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy. It was a long run project which started with Hernando Barragán's Master's thesis project in 2003, and in 2005 was extended by Massimo Banzi, David Mallis and David Cuartielles – other IDII students.

There exist many types of Arduino boards such as Arduino Nano, Arduino Micro, Arduino Due, LilyPad Arduino Board, Arduino Mega (R3) Board, Arduino Robot, and lots of other types; however, the one which is mostly used due to its reasonable price and practicality, is Arduino Uno, which was applied even in this experiment. Arduino Uno board depends on an ATmega328P based microcontroller. It consists of 14-digital I/O pins, where 6 – pins can be used as PWM (pulse width modulation outputs), 6-analog inputs, a reset button, a power jack, a USB connection, an In-Circuit Serial Programming header (ICSP), etc. This platform utilizes Arduino programming language (base on Wiring), and the Arduino Software (IDE), based on Processing. Arduino is used for different purposes such as hobbies, design, art, hacking, and many other creative ideas. Arduino is interactive with LEDs, PC, smartphones, TV, motors, and it goes as far as being the base for many electronics projects.

1.2.2 Arduino software

The integrated Development Environment (IDE) of Arduino is free and open source. It has a minimalist design, with only six headings on the menu bar, such as compile and check errors, upload codes to Arduino, create new sketch, open an existing sketch, etc. The programming language used in this IDE is a simplified version of C/C++ programming language, where we acknowledge two mandatory functions: the `setup()` and the `loop()` functions.

`Setup()` – Every Arduino sketch must have a `setup` function. This function defines the initial state of the Arduino upon boot and runs only once.

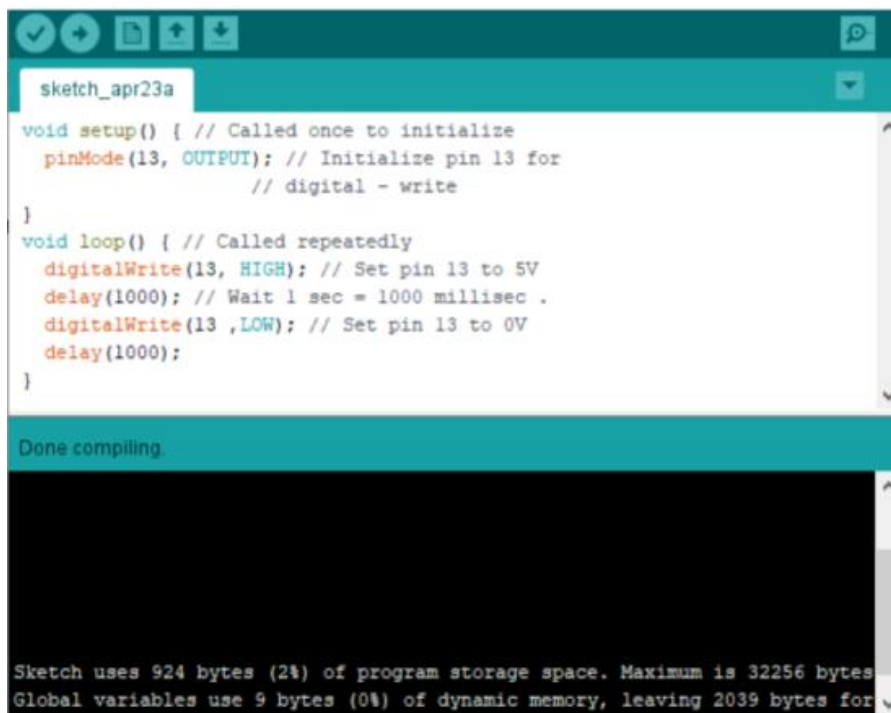
`Loop()` – The `loop` function executes once `setup()` is complete. It is the main function and as its name hints, it runs in a loop. Moreover, it describes the main logic of the circuit.

1.3 Equipment

- ◆ **Arduino Board** – reads the input and returns an output, signaled by the lighting of a LED.
- ◆ **Arduino IDE**
- ◆ **Breadboard** - solderless construction base used for developing electronic circuit and wiring. It consists of two areas called strips, which are separated from the middle portion. Bus strips are mainly used for power supply connections. Terminal strips are used mostly for electrical components.
- ◆ **Jumper Wires** – wires that have connector pins at each end.
- ◆ **LED** - a semiconductor diode which glows when a voltage is applied.
- ◆ **Resistor** – a passive two-terminal electrical component that limits or regulates the flow of electrical current in a circuit.
- ◆ **Multimeter** – instrument that measures electric current, voltage, and resistance over several ranges of value.
- ◆ **Potentiometer** – a device that measures the potential difference by comparison with a known voltage (a variable voltage divider).

1.4 Procedure and execution

Task 1.1



```
sketch_apr23a

void setup() { // Called once to initialize
  pinMode(13, OUTPUT); // Initialize pin 13 for
                      // digital - write
}
void loop() { // Called repeatedly
  digitalWrite(13, HIGH); // Set pin 13 to 5V
  delay(1000); // Wait 1 sec = 1000 millisec .
  digitalWrite(13 ,LOW); // Set pin 13 to 0V
  delay(1000);
}

Done compiling

Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256 bytes
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for
```

This is the code that was supposed to be shown to the TAs. The code above enabled the LED to blink every one second.

Task 1.2

In Tinkercad, Arduino Uno simulation, this was done by stopping the simulation from running. The simulation of Arduino Uno in Tinkercad made it significantly easy to pull out USB cable from the Arduino. This is illustrated in the figure below:

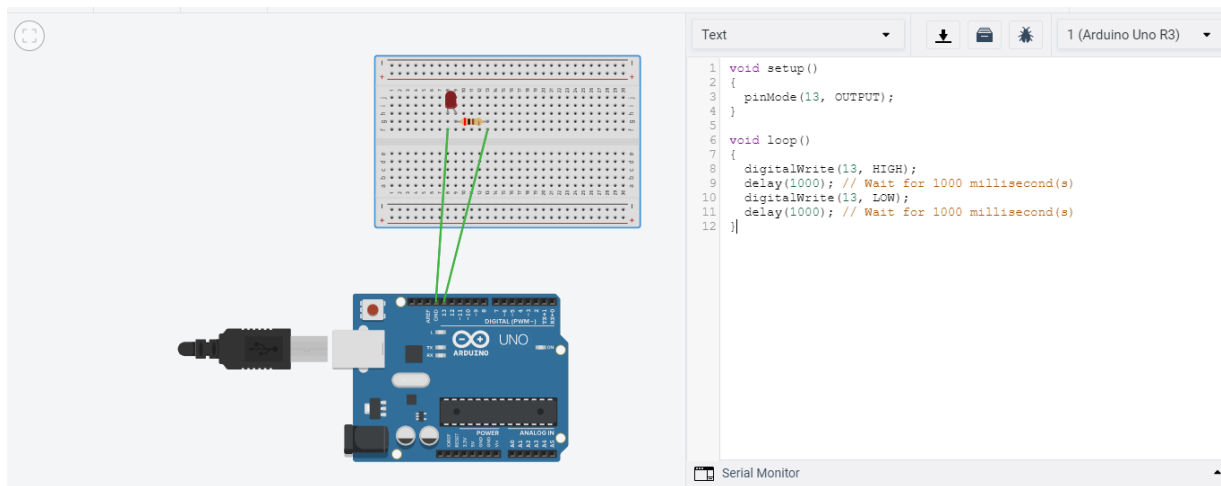
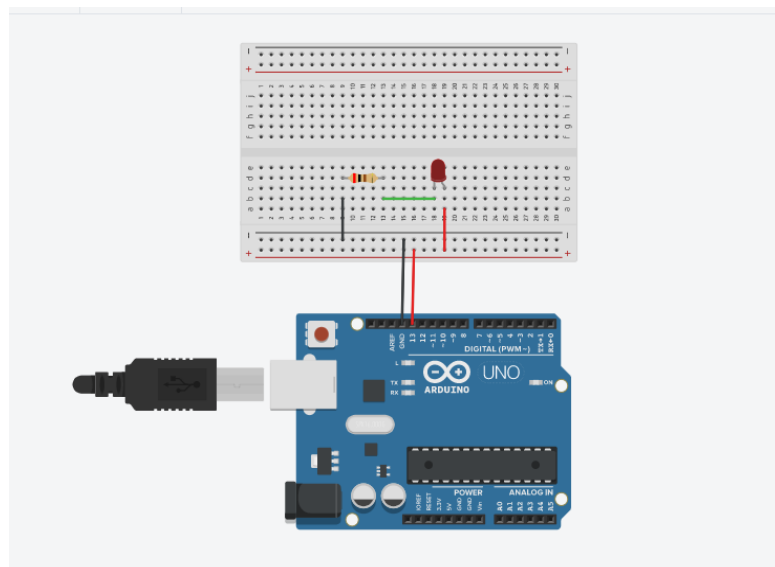


Figure 1: Circuit for task 1.2

As you can see in the second image, the light turned off after 1 second. That is because the light was lighting up every second.



Task 1.3

In this task the resistance of different carbon resistors was supposed to be obtained through the usage of the multimeter. However, since the work was done in a stimulation, then the resistances measured were first altered, and then estimated by the multimeter.

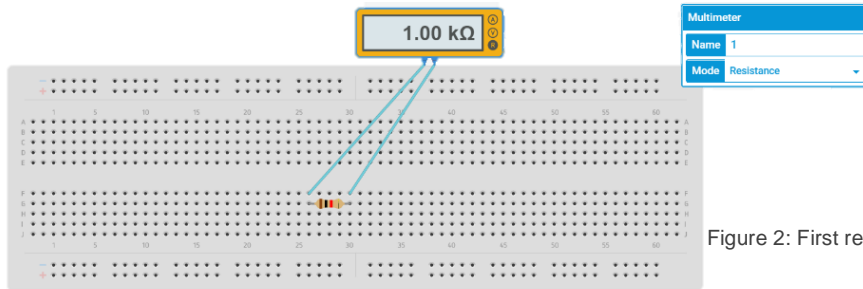


Figure 2: First resistor measured

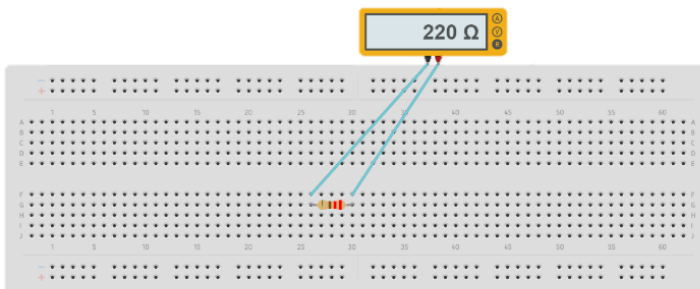


Figure 3: Second resistor measured

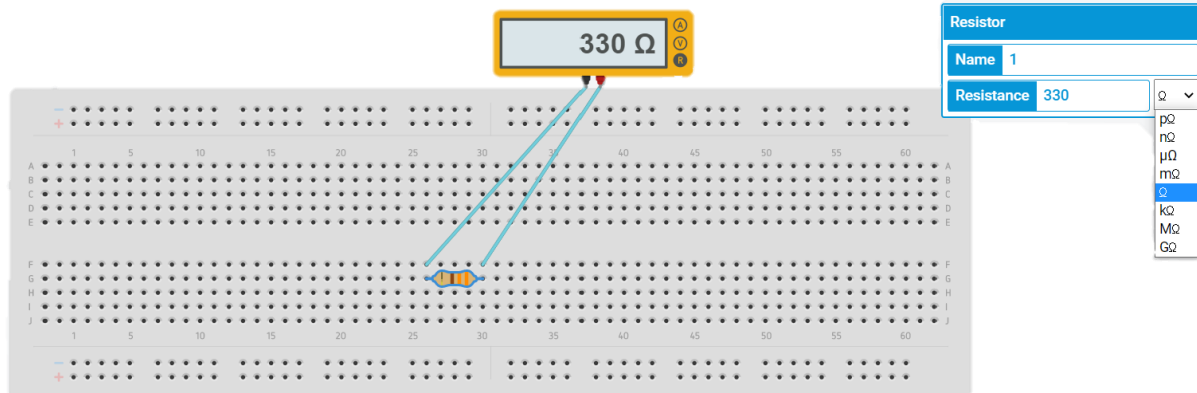


Figure 4: Third resistor measured and the resistance alternation in Tinkercad

Task 1.4

A voltage of 2.00V was measured when the voltmeter was connected to the circuit.

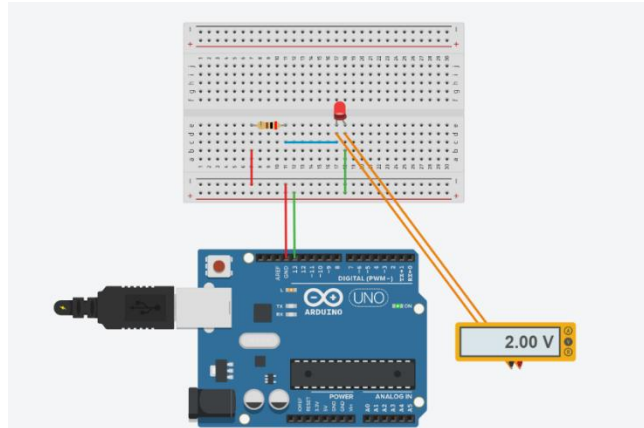


Figure 5: Measuring the voltage

Task 1.5

After adjusting the multimeter to the continuity test, we connected the leads of the multimeter to the two pins in the button. Without pressing the button, we heard nothing, but when we pressed the button we heard a loud, continuous beep, which indicated they were shorted.

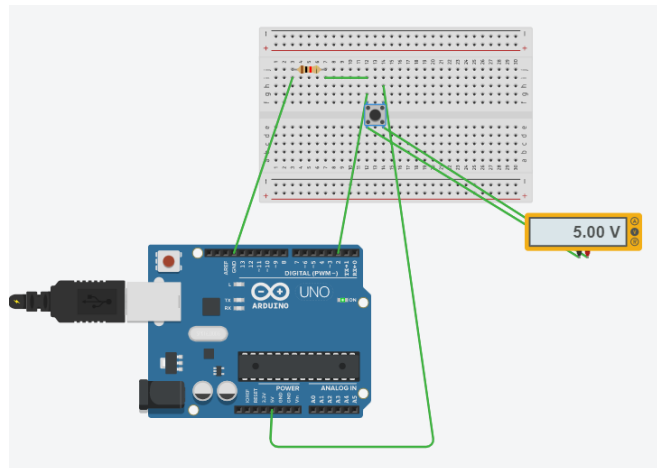


Figure 6: The Button

Task 1.6

After following the schematic from the manual, and uploading on Tinkercad the code provided in that same manual, the observation of the behavior of the built-in LED on Pin13 took place. Apparently, when the circuit is closed, but the button is opened, due to the lack of connection between the two legs of the push-button, the built-in LED does not ignite. However, the opposite happens when we press the push-button.

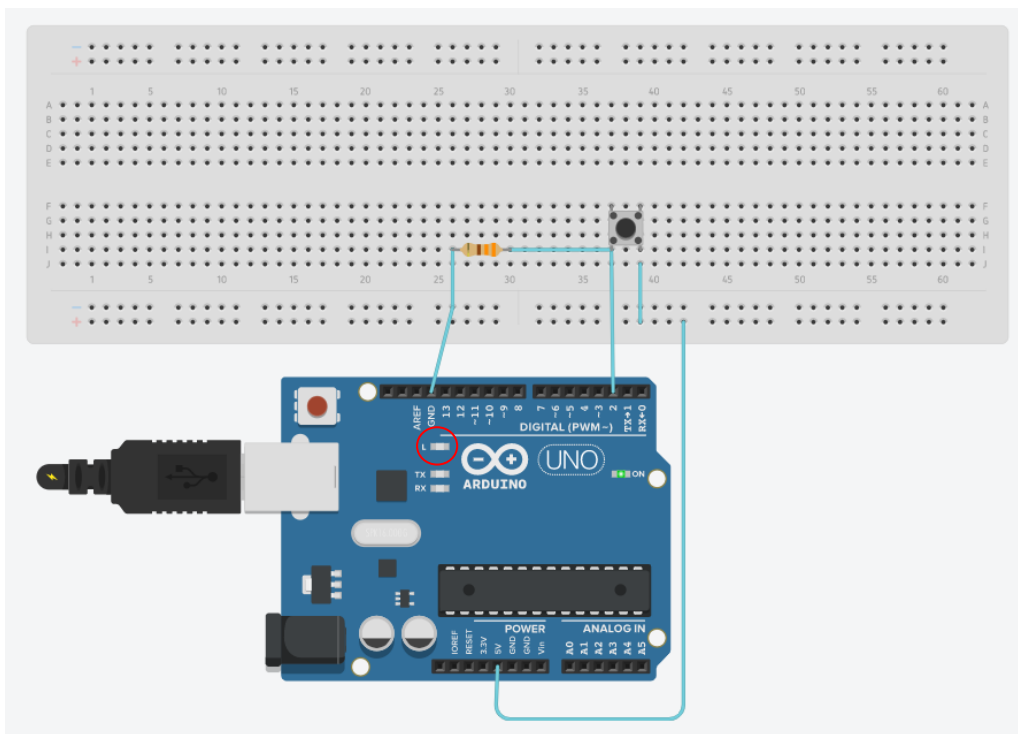


Figure 7: Unpressed button

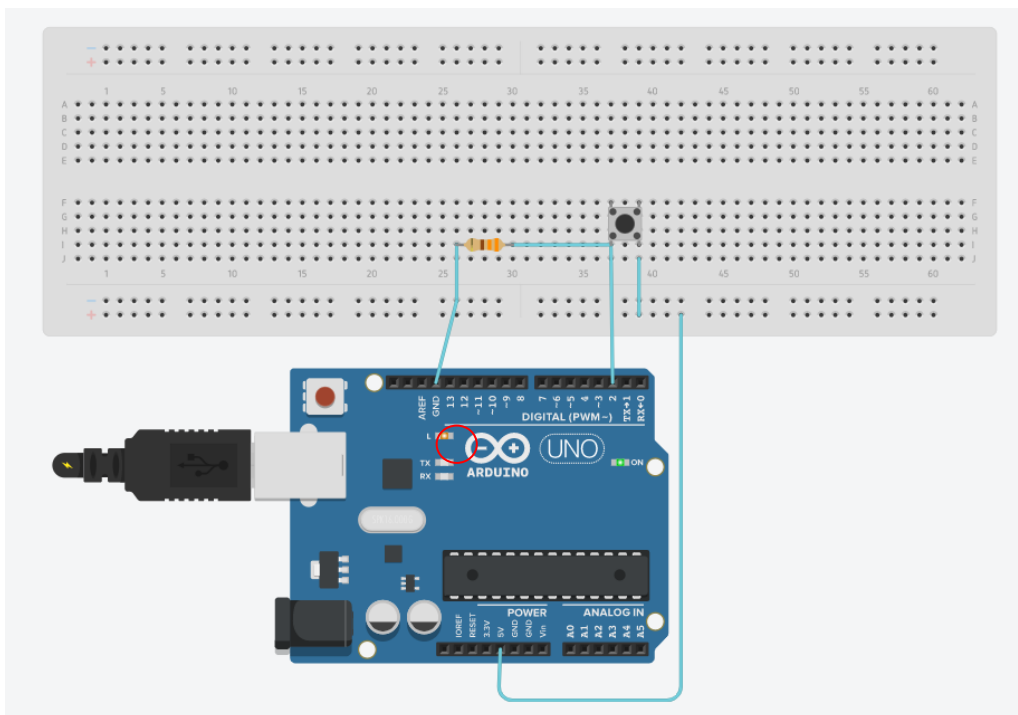


Figure 8: Pressed button

Task 1.7

The total resistance measured was 10.00 k Ω

Task 1.8

This is the circuit with the potentiometer and the voltmeter.

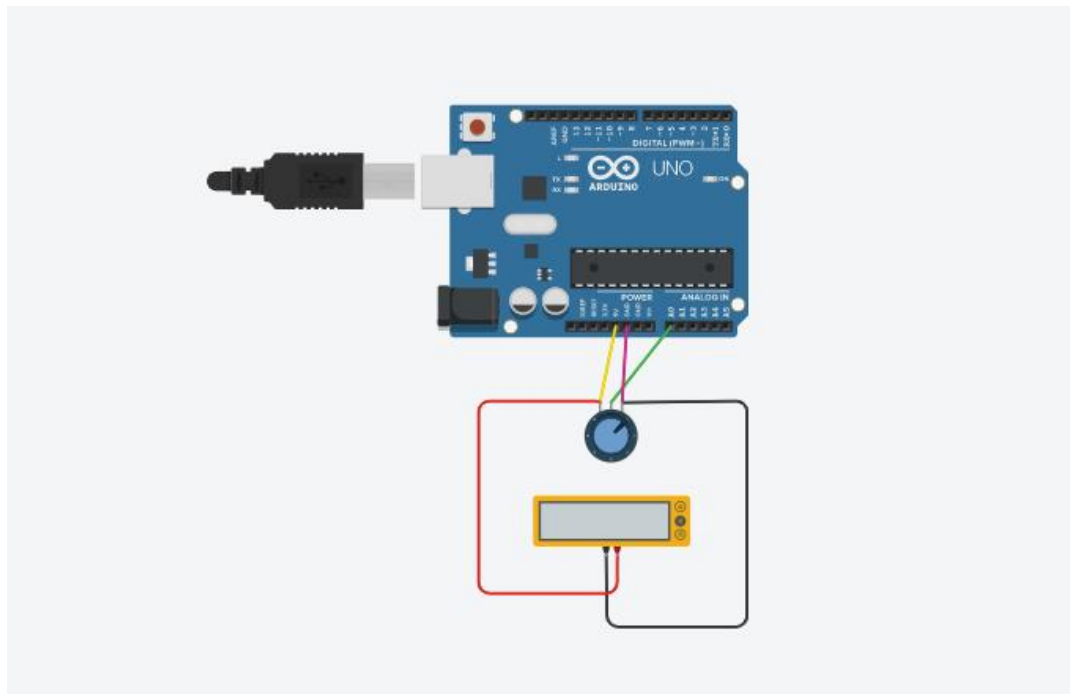


Figure 9: The circuit with the potentiometer and the voltmeter

This is what we observed in the serial monitor when we changed the resistance by turning the shaft of the potentiometer



1.5 Discussion/Conclusion

In conclusion, the experiments for this first lab were aimed to familiarize us with programming the Arduino. It can be observed that Arduino has many applications in the real world because of the sensors. Accurate measurements of resistance, voltage and distances can be calculated using the appropriate sensors, a multimeter and the Arduino itself. Procedures were followed and practiced to implement the programs for Arduinos and see their output. These evaluations were achieved by preparing the set-up in such a way that it allowed the investigation and the recording of the voltage, resistance, current flows and others. Then, the data was collected as output as the solutions for every task presented. Finally, the introduction to the software of Arduino and sample codes that consisted of two parts: void setup() and void loop() was successful and procedures and solutions for the task are depicted in the lab report.

Table of figures

Figure 1: Circuit for task 1.2	5
Figure 2: First resistor measured	1
Figure 3: Second resistor measured	1
Figure 4: Third resistor measured and the resistance alternation in Tinkercad	1
Figure 5: Measuring the voltage	2
Figure 6: The Button	2
Figure 7: Unpressed button	3
Figure 8: Pressed button	3
Figure 9: The circuit with the potentiometer and the voltmeter	4

References

<https://www.arduino.cc/en/guide/introduction>

<https://en.wikipedia.org/wiki/Arduino>

<https://www.elprocus.com/different-types-of-arduino-boards/>

<https://www.circuito.io/blog/arduino-code/#:~:text=What%20language%20is%20Arduino%3F,and%20compiled%20to%20machine%20language.>

Ph.D. Hu, Fangninv, Dr. Pathak, Kaustubh. Lab Manual, 2021.

McRoberts, Michael. Beginning Arduino. Apress, 2011.



JACOBS
UNIVERSITY

Jacobs University
CH-220-B Introduction to Robotics
Fangning Hu
Lab Session 2
19 April 2021
Campus Ring 1, 28759 Bremen
Group 3, Cohort B -Simulation
Bazl-e- Fatima
Blen Assefa
Era Gërbeshi

TABLE OF CONTENTS

Abstract	3
Theory.....	3
LDR - Light Dependent Resistor	3
Temperature sensors.....	4
Ultrasonic Distance Sensor.....	4
Servomotor.....	5
Processing.....	6
Execution and procedure.....	6
Task 2.1	6
Task 2.2.....	7
Task 2.3.....	8
Task 2.4.....	9
Task 2.5.....	9
Task 2.6.....	11
Task 2.7	11
Task 2.8.....	11
Task 2.9.....	12
Task 2.10	13
Task 2.11	13
Task 2.12	14
Discussion and Conclusion.....	15
References.....	15

ABSTRACT

In this experiment some additional electronic components of the Arduino Kit, such as LDR (Light Dependent Resistor), temperature sensor, ultrasonic distance sensor, and servomotor were inspected in more details. Another investigation was on Processing programming language.

THEORY

LDR - LIGHT DEPENDENT RESISTOR

An LDR or Light dependent Resistor is also known as a photo resistor, photocell, photoconductor, etc. LDR as the name states it is a special type of resistor that works on the photoconductivity principle. When light falls upon it, then the resistance changes. This resistance will decrease when it encounters incident light, and can go up to several megaohms in darkness. With such variation in resistance LDRs are easy to use in different electronic equipment, such as photographic light meters, fire or smoke alarms as well as burglar alarms, and they also find uses as lighting controls for street lamps; moreover, they can be used to detect nuclear radiation. LDRs are made from semiconductor materials like silicone, germanium, Ge, lead sulphide, PbS, indium antimonide, InSb, cadmium sulphide, CdS, although the use of these cells is now restricted in Europe due to environmental issues with the use of cadmium and lead. LDRs have high resistance because the vast majority of the electrons are locked into the crystal lattice and unable to move. But, as light falls on the semiconductor, the light photons get absorbed by the semiconductor lattice and some of their energy is transferred to the electrons. This energy is sufficient for the electrons to break free from the crystal lattice, which results to a smaller resistance from the LDR. Light dependent resistors fall into two types of categories:

Intrinsic photoresistors – which uses un-doped semiconductor materials including silicon or germanium. Photons that fall on the photoresistor excite the electrons, and the more electrons are liberated the greater the level of conductivity.

Extrinsic photoresistors – are manufactured from semiconductors of materials doped with impurities. In these types of photoresistors electrons need less energy to transfer to the conduction band because the energy gap decreases due to the impurities that create a new energy band above the existing valence band.

The symbol that represents the LDR in electronic circuits is based on the symbol of the resistor, but for more specification it shows some extra arrows, which indicate the light.



Figure 1: The symbol of a photoresistor

Source: <https://en.wikipedia.org/wiki/Photoresistor>



Figure 2: LDR

Source: <https://www.indiamart.com/proddetail/ldr-photoresistor-photo-light-sensitive-resistor>

TEMPERATURE SENSORS

There exist a wide variety of temperature sensors with different features, however, we would be describing the temperature sensor TMP36. So, the TMP36 is a low voltage, analog temperature sensor, with a centigrade precision sensor. It provides a voltage output that is linearly proportional to the Celsius temperature. This sensor uses a solid-state technique to determine the temperature, by using the fact that as the temperature increases, the voltage across a diode increases at a known rate. TMP36 does not require any external calibration. This temperature sensor has a temperature range from -40°C to 150° , a power supply fluctuating from 2.7V to 5.5V, an accuracy of $\pm 1^{\circ}$ (at 25°C) and $\pm 2^{\circ}$ (between -40°C to 125°C), and an output range from 0.1V to 2.0V. The temperature sensor has three pins. If the view is from the flat side, where there is some text written on, the left pin should be connected to 5V, the right pin to GND, and the middle pin contains the temperature-signal and hence it should be connected to A0. TMP36 is used in environmental control systems, fire alarms, for thermal protection, industrial process control, power system monitors, CPU thermal management, and so forth.

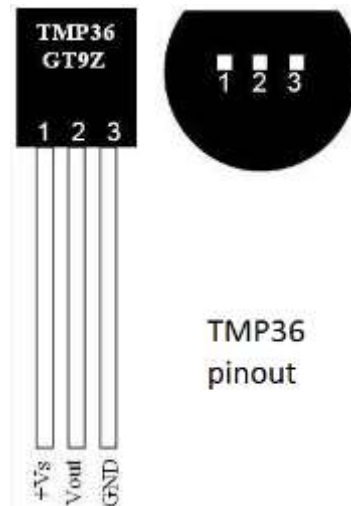


Figure 3: Temperature sensor

Source: <https://content.instructables.com/ORIG/F17/L3DF/IBW25LH3/F17L3DFIBW25LH3.png>

ULTRASONIC DISTANCE SENSOR

The HC-SR04 Ultrasonic Sensor is a sensor used for detecting the distance to an object using sonar. So, this detector uses non-contact ultrasound sonar to measure the distance to an object, and consists of two ultrasonic transmitters (basically speakers), a receiver, and a control circuit. The Ultrasonic Distance Sensor has only four pins: VCC (Power) pin, Trig (Trigger), Echo

(Receiver), and GND (Ground). This sensor works by sending a signal, a high frequency sound (40kHz), through the transmitter (trig pin), which bounces off any nearby solid objects, and the receiver listens for any return echo. That echo is then processed by the control circuit, which by calculating the time between the released sound to the perceived echo, it finds out the distance between the sensor and the reflecting object. The HC-SR04 sensor works best between 2cm – 400 cm within a 30° cone, and is accurate to the nearest 0.3cm.

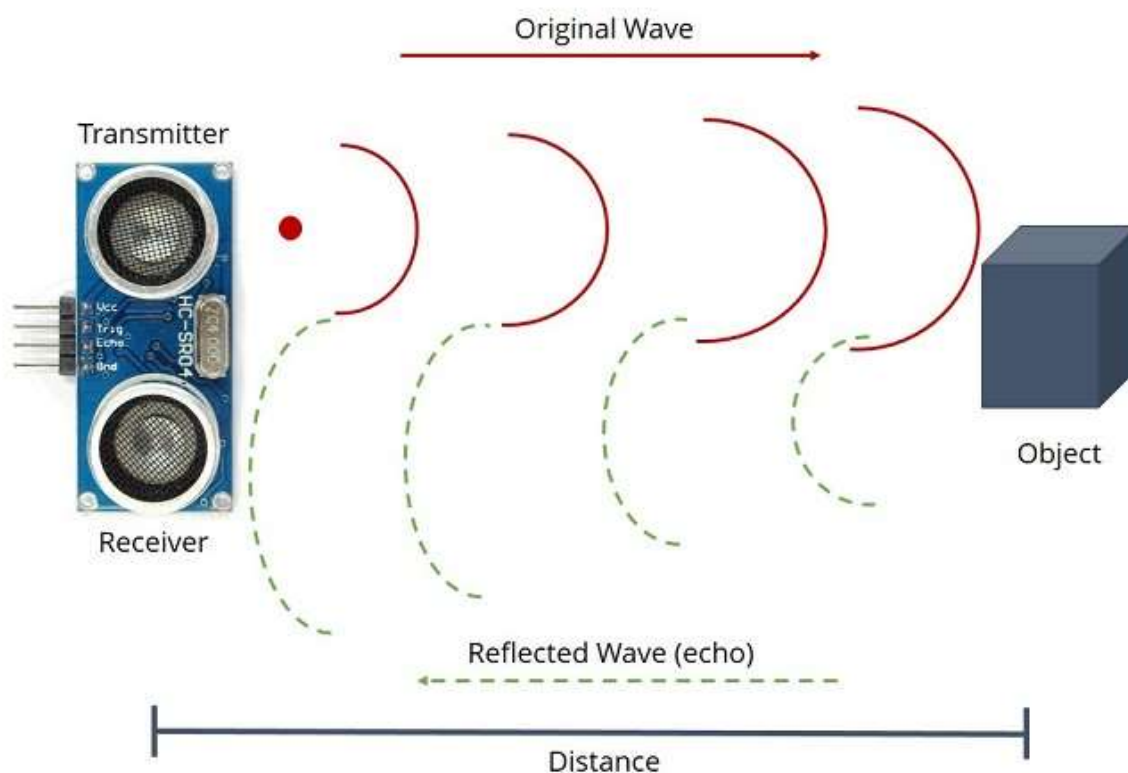


Figure 4: Ultrasonic sensor work principle

Source: <https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/>

SERVOMOTOR

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity, and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. Servomotors have a shaft that can rotate at a specific angle, while linear servos have an output rod that slides back and forth. Motors used in servomotors are either asynchronous motors, synchronous motors, or DC motors. The name servomotor comes from the Latin word “servus”, which means “server”. Small servomotors are widely used for remote controlled airplanes, robots, cars, and other toys, while servomotors which are bigger find

application in controlling valves in plants, powering steering in automobiles, and other large systems.

PROCESSING

Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual art. As an integrated development environment (IDE) and an open-source language, processing programming language has been specifically developed for the communities of visual design, new media art, and electronic art. The work on the development of processing was initiated in spring of 2001. The language got its basic source of development from the Design of Media in Numbers project. Later on, processing programming language became a source for the Arduino and Wiring projects. Processing uses Java language, with additional simplification such as additional classes and aliased mathematical functions and operations. It also provides a graphical user interface for simplification purposes. The IDE of processing has a Text Editor, together with a row of buttons at the top, basically the toolbar. Below the editor is the Message Area, and below that is stands the Console.

EXECUTION AND PROCEDURE

TASK 2.1

The following task was completed on Thinkercad. There, the light that was supposed to illuminate on the light dependent resistor, was adjusted through a slider. Meanwhile, the resistance displayed by the photoresistor was obtained with the help of a multimeter, which was connected to that LDR (photoresistor). The set-up for this task is presented on figure 5. On the other hand, the relation between the resistance of the photoresistor, and the percentage of the incident light is shown on Table 1. Another thing, which is worth mentioning is that this percentage is approximated and rounded, and not read anywhere precisely.

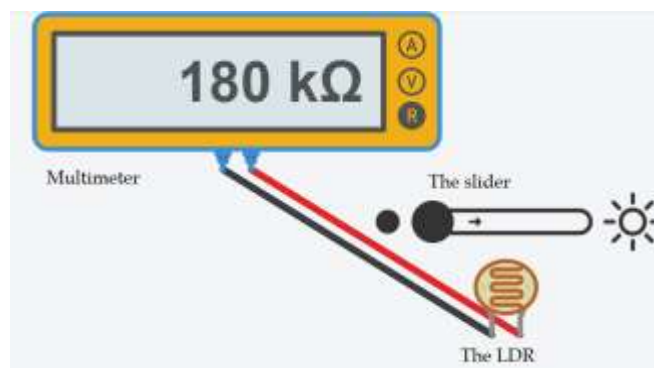


Figure 5: The set-up for measuring the resistance displayed by the LDR

Slider percentage	0%	25%	50%	75%	100%
Resistance	180 k Ω	1.70 k Ω	882 Ω	587 Ω	506 Ω

Table 1: The relation between the amount of light (in percentage) shining on the LDR, and the resistance displayed by this LDR

TASK 2.2

This is the circuit when the LDR is uncovered (as shown by the brightness bar in the figure), as you can see the LED light doesn't turn on.

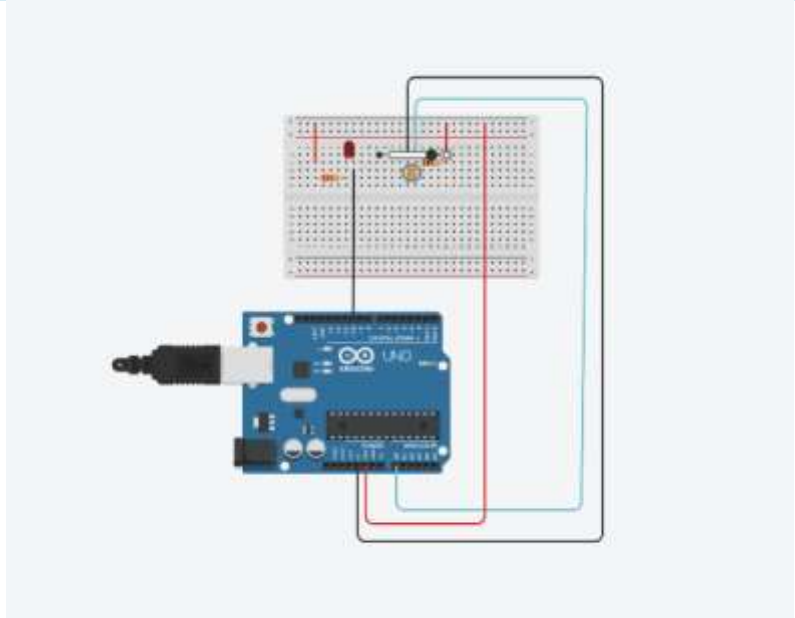


Figure 6: Circuit where the LDR is uncovered

This is the circuit when the LDR is covered (as shown by the brightness bar in the figure above), as you can see the LED light is turned on.

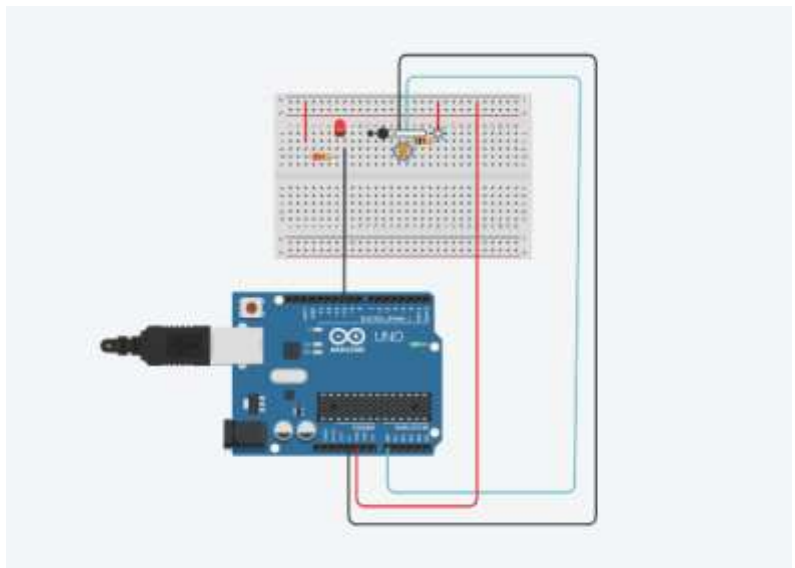


Figure 7: Circuit where the LDR is covered

The code used for this task is:

```
1 // C++ code
2 //
3
4 int input_pin = A0;
5 int LED = 10;
6 int sensor_sample = 0;
7
8 void setup()
9 {
10   Serial.begin(9600);
11   pinMode(LED, OUTPUT);
12 }
13
14 void loop()
15 {
16   sensor_sample = analogRead(input_pin);
17   Serial.print("Sensor value = ");
18   Serial.println(sensor_sample);
19   if(sensor_sample < 650){
20     digitalWrite(LED, HIGH);
21   }else{
22     digitalWrite(LED, LOW);
23   }
24   delay(100); // Wait for 100 millisecond(s)
25 }
```

Figure 8: Code for testing the LDR

TASK 2.3

In the figure you will see the circuit built for measuring the average temperature. In order to output 10 different temperature values, we needed to add the code to the Arduino and get an output of the temperature values and the standard deviation of each value.

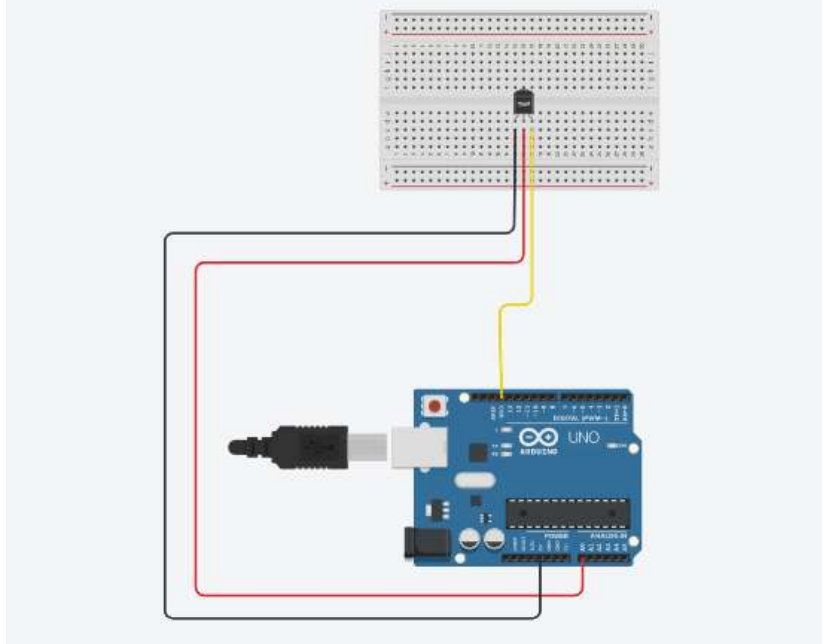


Figure 9: Circuit for measuring the temperature

The code for seeing the output of the circuit is:

```
1 // C++ code
2 //
3 int TMP36 = A1;
4 int temperature = 0;
5 int wait_ms = 20;
6 #define NR_SAMPLES 10
7 int samples[NR_SAMPLES];
8
9 void setup() {
10   Serial.begin(9600);
11 }
12
13 void loop() {
14   float sum = 0.0;
15   // Maps values from range [0, 410] to [1]
16   for (int i = 0; i < NR_SAMPLES; ++i) {
17     samples[i] = map(analogRead(TMP36), 0, 410, -50, 150);
18     sum += samples[i];
19     delay(wait_ms);
20   }
21   float mean = sum/NR_SAMPLES;
22   float sum_square_deviation = 0.0;
23   for (int i = 0; i < NR_SAMPLES; ++i) {
24     sum_square_deviation += (samples[i] - mean) * (samples[i] - mean);
25   }
26   float standard_deviation = sqrt(sum_square_deviation/NR_SAMPLES);
27   Serial.print("Mean: ");
28   Serial.print(mean, 3);
29   Serial.print("C, ");
30   Serial.print(standard_deviation);
31 }
```

Figure 10: Code for measuring the temperature

The output on the serial monitor is shown in the figure below.

Serial Monitor	
mean : -40.400 C,	std: 0.92
mean : -39.000 C,	std: 0.00
mean : -31.400 C,	std: 4.41
mean : -10.800 C,	std: 6.87
mean : 7.800 C,	std: 8.51
mean : 32.700 C,	std: 7.48
mean : 54.000 C,	std: 6.00
mean : 75.000 C,	std: 8.00
mean : 100.100 C,	std: 9.28
mean : 122.800 C,	std: 1.47

Figure 11: Serial Monitor output for seeing the output of the measured temperature

TASK 2.4

The temperature sensor of type TMP36 outputs 0 – 2 V with temperatures between - 50°C and 150°C – on a 200°C span. Since the Arduino, takes values between 0 – 5 V, the value 410 is used for the sake of calibrating and obtaining the right evaluations. So:

$$\left(\frac{2}{5}\right) \cdot 1024 \approx 410$$

410/200 = about two values per °C.

TASK 2.5

The experiment is shown in the setup below.

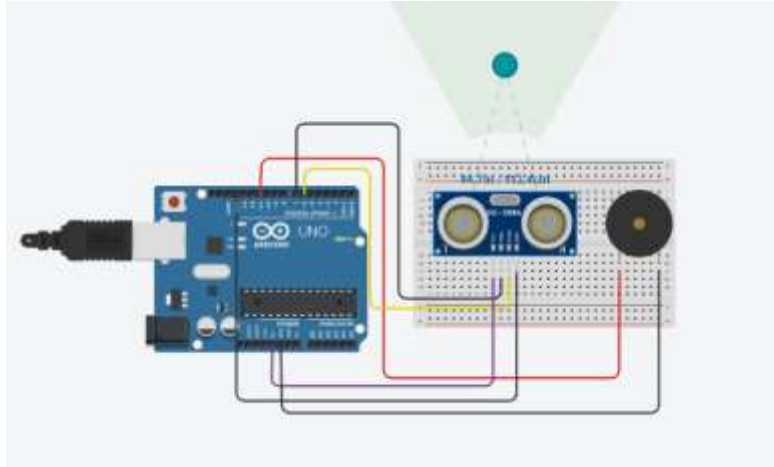


Figure 12: Circuit with piezo speaker beeps with some distance

With the following code:

```
1 // C++ code
2 //
3 const int trig = 7;
4 const int echo = 6;
5
6 void setup()
7 {
8   pinMode(trig, OUTPUT);
9   pinMode(echo, INPUT);
10  digitalWrite (trig, LOW);
11  Serial.begin(9600);
12 }
13
14 void loop()
15 {
16   digitalWrite(trig, HIGH);
17   delayMicroseconds(15);
18   digitalWrite(trig, LOW);
19   long duration = pulseIn(echo, HIGH);
20
21   const long vsound = 340;
22   long dist = (duration / 2L) * vsound / 10000L;
23
24   if( dist > 500L || dist < 2L )
25   {
26     Serial.println("Invalid range!");
27     digitalWrite(11, LOW);
28   }else{
29     Serial.print(dist);
30     Serial.println("cm");
31     if(dist < 100){
32       digitalWrite(11, HIGH);
33     }else{
34       digitalWrite(11, LOW);
35     }
36   }
37   delay(1000);
38 }
```

Figure 13: code for the circuit with piezo speaker with some distance

TASK 2.6

Using the Arduino Uno, we were able to see if it is possible to plug in different pins to see whether PWM software programmed is supported by them or not, was implemented. This task also required us to use Servo. Servo is the motor with a function which takes in an input such that it rotates its axis to a certain degree specified by the program.

The 2 PWM pins that are disabled while using the library on an Arduino are pins 9 and 10, even if there is a servo in these pins. If you are using the Mega, up to 12 servos can be used without interfering with the PWM functionality, the use of motors 12 to 23 will disable pins 11 and 12.

TASK 2.7

The code used in the preceding task was changed in such manners that after the servo, with the shaft attached to it, went through the previous path, it returned back to the initial 0° position. This was achieved by adding `s.write(0); delay(3000); exit(0);` to the loop function as represented below:



```
1 #include <Servo.h>
2 Servo s;
3 void setup() {
4   s.attach(8);
5 }
6 void loop() {
7   s.write(0);
8   delay(3000);
9   s.write(45);
10  delay(3000);
11  s.write(90);
12  delay(3000);
13  s.write(135);
14  delay(3000);
15  s.write(180);
16  delay(3000);
17  //Return to zero degree
18  s.write(0);
19  delay(3000);
20  exit(0);
21 }
```

Figure 14: The code for turning the shaft to the starting point – 0 degree

TASK 2.8

This is the code we used to make the servo transition smoothly from 0 to 180 degrees, and back in a continuous cycle.

A cycle of the servo from 0 to 180 degrees and back is created by changing the given code. Then the period of the rotation is set to 6 seconds by the mathematical additions to the code.

```


1 // C++ code
2 //
3 #include <Servo.h>
4
5 Servo s; // create servo object ;
6 int pos = 0;
7
8 void setup()
9 {
10   s.attach(8); //control signal on pin 0
11 }
12
13 void loop()
14 {
15   for(pos = 0; pos <= 180; pos += 1){
16     s.write(pos);
17     delay(6);
18   }
19   for(pos = 180; pos >= 0; pos -= 1){
20     s.write(pos);
21     delay(6);
22   }
23 }

```

Figure 15: The code for the servo

TASK 2.9

A 2D car was drawn by using the Processing programming language. The code and the image of the car are displayed below.



```

1 void setup(){
2   size(700,600);
3   smooth();
4 }
5
6 void draw(){
7   strokeWeight(30);
8   ellipse(150,350,50,50);
9   ellipse(470,350,50,50);
10  strokeWeight(3);
11  ellipse(230,270, 20, 20);
12
13  strokeWeight(4);
14  line(190,350,450,350);
15  line(45,350,110,350);
16  line(490,350,650,350);
17  line(45,350,45,250);
18  line(45,250,85,250);
19  line(85,250, 530,250);
20  line(85,250,200,185);
21  line(200,185,400,185);
22  line(350,185,350,350);
23  line(400,185,525,250);
24  curve(10,260,200,185,198,250,10,200);
25  line(198,250, 198,350);
26  curve(30,340,525,250,650,350,70,600);

```



Figure 16: 2D car drawn with Processing.

TASK 2.10

The car was modified using the background(), fill(), and stroke() function. Apparently, these function work properly when are placed before the component which they try to influence.

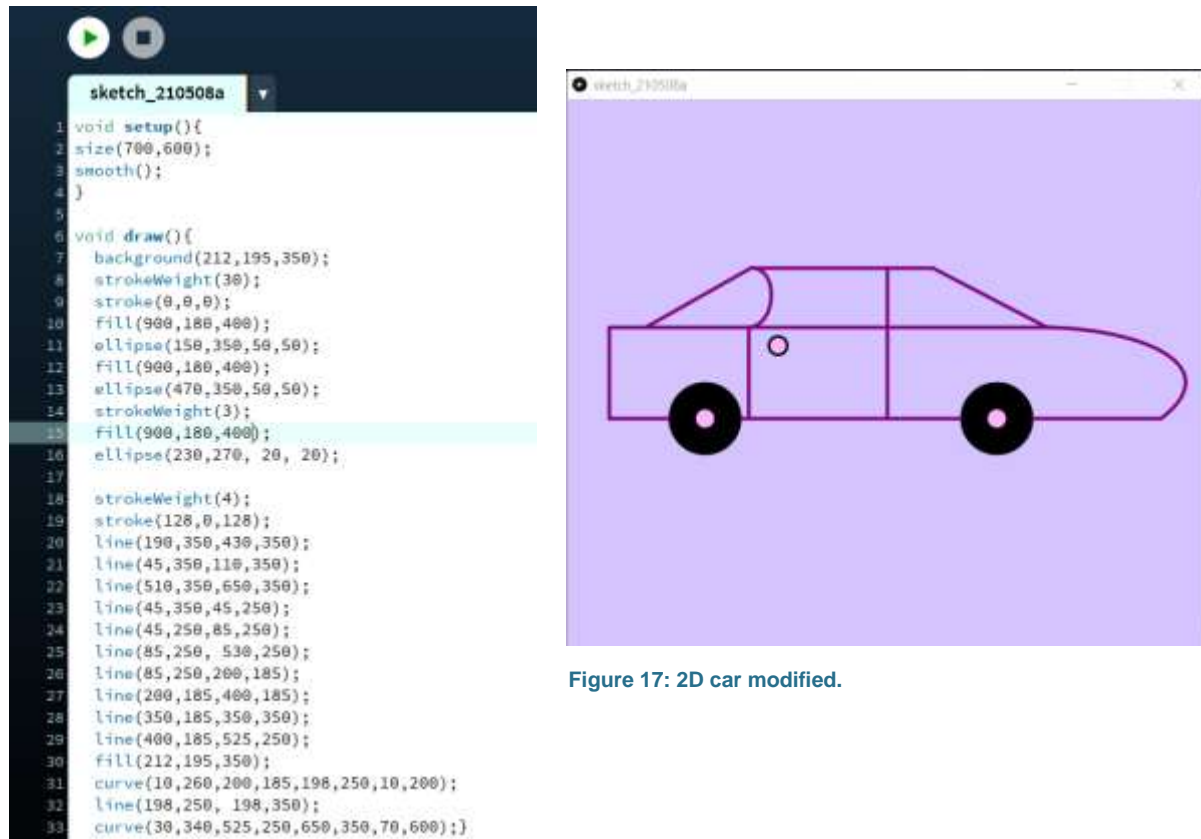


Figure 17: 2D car modified.

TASK 2.11

In the lab manual, sample code given to us. It was copied and pasted into the editor and then ran to observe the right program.



Figure 18: Code for processing

The observations can be seen below.

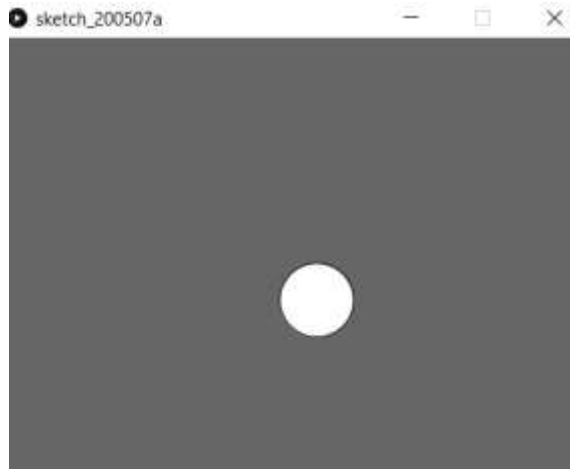


Figure 19: Output of the code compiled

Afterwards the background function was moved from the setup function to the draw function and the program was run once more to see the differences

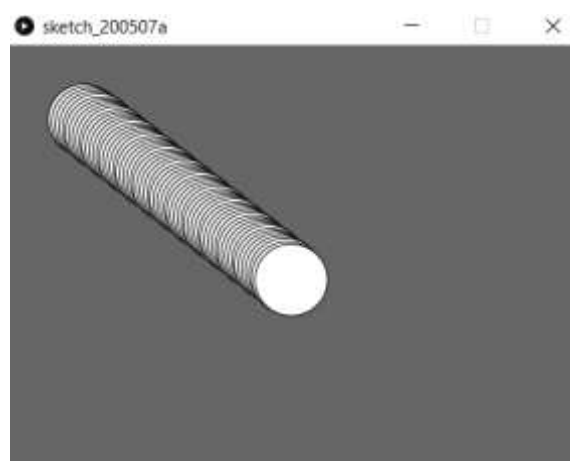


Figure 20: Output of the code after removing background function

TASK 2.12

For the sample code to run, there needed to be a link between the arduino board and processing through the use of a port. The position of the mouse in the direction of the x-axis is inserted as a byte to arduino by processing. Arduino reads the received bytes and sets it as the power variable delivered to power the light. Less intense light is represented by a lower value and higher intense light by a higher value.

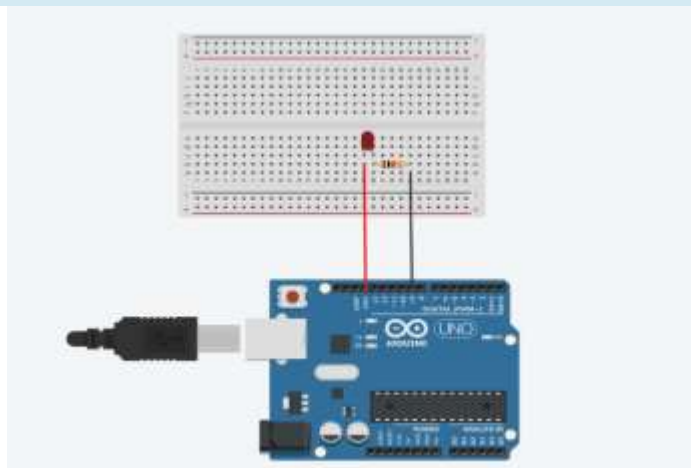


Figure 21: Circuit diagram when the Arduino board is connected to the processing through the use of port

DISCUSSION AND CONCLUSION

Overall, the idea behind the experiments for this lab was to practice the connections of several different sensors to the Arduino and see their output. These evaluations were achieved by preparing the set-up in such a way that it allowed the investigation and the recording of the voltage, resistance, current flows and others. Then, the data was collected as output as the solutions for every task presented.

REFERENCES

- adafruit. (2021, April 27). Retrieved from <https://learn.adafruit.com/tmp36-temperature-sensor>
- CLEVERism. (n.d.). Retrieved from CLEVERism: <https://www.cleverism.com/skills-and-tools/processing-programming-language/>
- Components101. (2018, March 13). Retrieved from <https://components101.com/sensors/tmp36-temperature-sensor>
- electronicsforu. (2020, October 15). Retrieved from <https://www.electronicsforu.com/resources/ldr-light-dependent-resistors-basics>
- electronicsnotes. (n.d.). Retrieved from electronicsnotes: https://www.electronics-notes.com/articles/electronic_components/resistors/light-dependent-resistor-ldr.php
- instructables circuits. (2020). Retrieved from instructables: <https://www.instructables.com/How-to-use-the-TMP36-temp-sensor-Arduino-Tutorial/>
- Ph.D. Fangning Hu, D. K. (n.d.). *Introduction to Robotic and Intelligent Systems Lab*. Bremen: Jacobs University.
- PiBorg. (n.d.). Retrieved from PiBorg: <https://www.piborg.org/sensors-1136/hc-sr04>
- Processing. (n.d.). Retrieved from Processing: <https://processing.org/>
- Processing. (n.d.). Retrieved from Wikipedia: [https://en.wikipedia.org/wiki/Processing_\(programming_language\)](https://en.wikipedia.org/wiki/Processing_(programming_language))
- RandomNerdTutorials. (2013). Retrieved from <https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/>
- WatElectronics. (n.d.). Retrieved from WatElectronics.com: <https://www.watelectronics.com/light-dependent-resistor-ldr-with-applications/>
- Wikipedia. (n.d.). Retrieved from Wikipedia: <https://en.wikipedia.org/wiki/Photoresistor>
- Wikipedia. (2021, March 21). Retrieved from Wikipedia: <https://en.wikipedia.org/wiki/Servomotor#:~:text=A%20servomotor%20is%20a%20rotary,a%20sensor%20for%20position%20feedback.>



Jacobs University
CH-220-B Introduction to Robotics
Fangning Hu
Lab Session 3
8 May 2021
Campus Ring 1, 28759 Bremen
Group 3, Cohort B -Simulation
Bazl-e- Fatima
Blen Assefa
Era Gërbeshi

Contents

Abstract.....	3
Theory	3
3D Transforms in Processing.....	3
Accelerometer	3
Execution and Procedure.....	4
Equipment.....	4
3D Transforms.....	4
<i>Task 3.1</i>	4
Acceleration for Finding Orientation	5
<i>Task 3.2</i>	5
<i>Task 3.3</i>	6
Discussion and conclusion	7
References	8

Abstract

Throughout this experiment the Processing programming language was inspected more thoroughly, together with the work principles of accelerometer.

Theory

3D Transforms in Processing

PDE has five render modes (Rendering or image synthesis is the process of generating a photorealistic or non-photorealistic image from 2D or 3D model by means of a computer program): the default renderer, P2D, P3D, PDF, and SVG. For using a non-default renderer, it is required to add the specific renderer into the `size()` function. Each one of the renders employs different libraries for drawing shapes, setting colors, displaying text, etc.

The xyz coordinate-system (CS) in Processing unfortunately is a non-standard left-handed CS. In the three-dimensional space of Processing the positive y-axis is facing down, meanwhile, the positive x-axis is directed to the right of the viewer, and the positive z-axis is coming off the screen. For specifying the 3D coordinates for shapes in Processing, the function `translate()` is used. This function moves the entire space and not just the drawn object, which makes the movement of larger objects, made by multiple components, more efficient. Other transforms such as scaling and rotating are managed by other functions.

Accelerometer

An accelerometer is a tool that measures proper acceleration. Acceleration is the rate of change of velocity. Accelerometers measure in meters per second (m/s^2) or in G-forces (g). A single G-force on Earth is equivalent to 9.8 m/s^2 , however, this does vary slightly with elevation. Accelerometers are useful for recording variations in systems or for detecting orientation. The accelerometer works by using an electromechanical sensor that is designed to measure either static or dynamic acceleration. Static acceleration is caused by forces such as gravity or friction. Dynamic acceleration, on the other hand, is caused by movement, vibration, or shock. Accelerometers can detect the acceleration on one, two, or three axes. They are made of capacitive plates. Some of which are fixed while the others are attached to a spring, which is responsible for catching the acceleration caused by the action of forces on it. As the plates move, the capacitance between them changes, from where the acceleration can be determined.

The accelerometer used in this experiment is the MMA8451, which is a miniature little accelerometer. It is designed for use in phones, tablets, smart watches, and more. It is quite precise, with a built in 14-bit ADC. This accelerometer has also a built-in orientation detector, which communicates over 12 C. MMA8451 accelerometer has a wide usage range, from $\pm 2\text{g}$ up to $\pm 8\text{g}$ yet is easy to use with Arduino or another microcontroller.

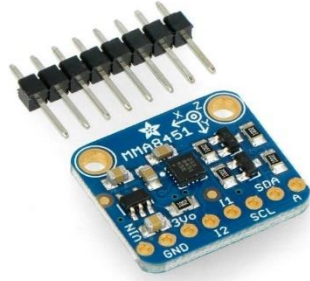


Figure 1: MMA8451 accelerometer

Source: <https://www.google.com/imghp?hl=en>

Execution and Procedure

Equipment

- Arduino Uno
- Breadboard
- Arduino software
- Jumper Cables
- Adafruit MMA8451 Accelerometer Breakout
- Processing IDE

3D Transforms

Task 3.1

The command `rotateX()` rotates the object around the x-axis, and takes the angle of rotation in radians, as a parameter. The command `rotateY()` does the same, but only in the y direction. It also uses the angle of rotation, expressed in radians, as its parameter. Both of these commands are used together with `mouseDragged()` function, which translates the recorded movement of the mouse into angle values. These angles then are used as the input of the previous mentioned commands.

Acceleration for Finding Orientation

Task 3.2

After the setup of the circuit, the code was uploaded taken from the manual. After that the breadboard was tilted in different directions and the directions of the x, y and z axis were observed. The results are shown below. There seem to be 8 possible orientation outputs.

```
Arduino /dev/cu.usbmodem14
16:54:32.991 -> X: -1562 Y: -2058 Z: 3292
16:54:32.991 -> X: -3.75 Y: -4.90 Z: 7.87 m/s^2
16:54:33.062 -> Landscape Left Front
16:54:33.062 ->
16:54:33.518 -> X: -1342 Y: -3342 Z: 1944
16:54:33.518 -> X: -3.19 Y: -7.98 Z: 4.64 m/s^2
16:54:33.591 -> Portrait Up Front
16:54:33.591 ->
16:54:34.040 -> X: -1936 Y: -3626 Z: 1224
16:54:34.040 -> X: -4.71 Y: -8.64 Z: 2.99 m/s^2
16:54:34.108 -> Portrait Up Front
16:54:34.108 ->
16:54:34.555 -> X: -2076 Y: -3492 Z: 190
16:54:34.589 -> X: -4.69 Y: -8.37 Z: 0.49 m/s^2
16:54:34.624 -> Portrait Up Front
16:54:34.624 ->
16:54:35.072 -> X: -2102 Y: -3518 Z: -1134
16:54:35.108 -> X: -5.01 Y: -8.35 Z: -2.71 m/s^2
16:54:35.144 -> Portrait Up Back
16:54:35.144 ->
16:54:35.595 -> X: -2050 Y: -2580 Z: -2604
16:54:35.633 -> X: -4.95 Y: -6.22 Z: -6.22 m/s^2
16:54:35.666 -> Portrait Up Back
16:54:35.666 ->
16:54:36.124 -> X: -1988 Y: -3016 Z: -2182
16:54:36.160 -> X: -4.70 Y: -7.25 Z: -5.24 m/s^2
16:54:36.194 -> Portrait Up Back
16:54:36.194 ->
16:54:36.652 -> X: -1674 Y: -2788 Z: -2138
16:54:36.688 -> X: -4.00 Y: -6.70 Z: -5.15 m/s^2
16:54:36.723 -> Portrait Up Back
16:54:36.723 ->
16:54:37.184 -> X: -1686 Y: -2910 Z: -2206
16:54:37.184 -> X: -4.02 Y: -6.97 Z: -5.27 m/s^2
16:54:37.218 -> Portrait Up Back
16:54:37.253 ->
16:54:37.680 -> X: -1726 Y: -2812 Z: -2254
16:54:37.713 -> X: -4.12 Y: -6.71 Z: -5.36 m/s^2
16:54:37.750 -> Portrait Up Back
16:54:37.786 ->
16:54:38.219 -> X: -1672 Y: -2972 Z: -2346
16:54:38.256 -> X: -4.05 Y: -7.10 Z: -5.66 m/s^2
16:54:38.293 -> Portrait Up Back
16:54:38.293 ->
```

Task 3.3

In these tasks, the code that was given beforehand was required to be changed in a way to make it sufficient enough to meet the requirements that are asked to find solutions for problems. The code adjusted is as follows:

```
sketch_210509a
1 import processing.serial.*;
2
3 int lf = 10; //Linefeed in ASCII
4
5 String receivedString = null;
6 Serial myPort; // Serial port you are using
7 float[] acceleration = {0., 0., 0.};
8 float pitch, roll;
9
10 void setup() {
11   size(640, 360, P3D);
12   noStroke();
13   colorMode(RGB, 1);
14   myPort = new Serial(this, "/dev/tty.usbmodem14201", 9600);
15   // Replace "COM1" by your Arduino port.
16   myPort.clear();
17 }
18
19 void draw_rgb_cube () {
20   background (0.4, 0.4, 0.4);
21   scale(100);
22   // Note the color convention: XYZ = RGB // Negative XYZ = light RGB
23   beginShape(QUADS);
24   // Front (As +Z is coming out of the screen) face: Red
25   fill(0, 0, 1);
26   vertex(-1, 1, 1);
27   fill(0, 0, 1);
28   vertex( 1, 1, 1);
29   fill(0, 0, 1);
30   vertex( 1, -1, 1);
31   fill(0, 0, 1);
32   vertex(-1, -1, 1);
33   // Right face (As +X is on the right): White
34   fill(1, 0, 0);
35   vertex( 1, 1, 1);
36   vertex( 1, 1, 1);
37   fill(1, 0, 0);
38   vertex( 1, 1, -1);
39   fill(1, 0, 0);
40   vertex( 1, -1, -1);
41   fill(1, 0, 0);
42   vertex( 1, -1, 1);
43   // Back face (-Z): Light Blue
44   fill(0.5, 0.5, 1);
45   vertex( 1, 1, -1);
46   fill(0.5, 0.5, 1);
47   vertex(-1, 1, -1);
48   fill(0.5, 0.5, 1);
49   vertex(-1, -1, -1);
50   fill(0.5, 0.5, 1);
51   vertex( 1, -1, -1);
52   // Left face (-X): Pink
53   fill(1, 0.5, 0.5);
54   vertex(-1, 1, -1);
55   fill(1, 0.5, 0.5);
56   vertex(-1, 1, 1);
57   fill(1, 0.5, 0.5);
58   vertex(-1, -1, 1);
59   fill(1, 0.5, 0.5);
60   vertex(-1, -1, -1);
61   // Bottom face (+Y is pointing down the screen): Green
62   fill(0, 1, 0);
63   vertex(-1, 1, -1);
64   fill(0, 1, 0);
65   vertex( 1, 1, -1);
66   fill(0, 1, 0);
67   vertex( 1, 1, 1);
68   fill(0, 1, 0);
```

```

69 vertex(-1, 1, 1);
70 // Top face (-Y): Light Green
71 fill(0.5, 1, 0.5);
72 vertex(-1, -1, -1);
73 fill(0.5, 1, 0.5);
74 vertex(1, -1, -1);
75 fill(0.5, 1, 0.5);
76 vertex(1, -1, 1);
77 fill(0.5, 1, 0.5);
78 vertex(-1, -1, 1);
79 endShape();
80 }
81
82 void draw () {
83   while (myPort.available() > 0) {
84     receivedString = myPort.readStringUntil(lf);
85     if (receivedString != null) {
86       boolean newSample= parseStringForAccelerations(receivedString);
87       if (newSample) {
88         print("Got acceleration: ");
89         print(acceleration[0]);
90         print(", ");
91         print(acceleration[1]);
92         print(", ");
93         print(acceleration[2]);
94         println(".");
95         // fill-in
96         float acc_norm = sqrt((acceleration[0]*(acceleration[0])+(acceleration[1]*(acceleration[1])+(acceleration[2]*(acceleration[2])));
97         float ax= acceleration[0]/acc_norm;
98         float ay= acceleration[1]/acc_norm;
99         float az= acceleration[2]/acc_norm;
100        // Fill -in:
101        pitch= atan(ax/az);
102        float sin_r=-ay;
103
104        float cos_r=ax*sin(pitch)+ax*cos(pitch);
105        roll=atan2(sin_r,cos_r);
106        //translate (
107        translate(width /2.0, height /2.0, -50);
108        rotateX(roll);
109        rotateY(pitch);
110        draw_rgb_cube();
111      }
112    }
113    myPort.clear();
114  }
115
116  boolean parseStringForAccelerations(String str) {
117    boolean sampleComplete= false;
118    String[] substrings= str.split(" ");
119    if (substrings[0].equals("X:")) {
120      acceleration[0]= float(substrings[1]);
121    } else if (substrings[0].equals("Y:")) {
122      acceleration[1]= float(substrings[1]);
123    } else if (substrings[0].equals("Z:")) {
124      acceleration[2]= float(substrings[1]);
125      sampleComplete= true;
126    } else {
127      print("Status message: ");
128      println(receivedString); // Prints status messages
129    }
130    return sampleComplete;
131  }
132

```

Discussion and conclusion

Throughout this experiment the work of accelerometer was observed, and the representation of the Cartesian space directions in the Processing programming language. All in all, the tasks were managed with the help of simulation as well as with the usage of hardware. In conclusion, the tasks that required us to fulfill the general and specific objectives of this lab were successfully done.

References

Accelerometer Basics. (n.d.). Retrieved from sparkfun: <https://learn.sparkfun.com/tutorials/accelerometer-basics/all>

Omega. (2018, August 28). Retrieved from Omega.org: <https://www.omega.com/en-us/resources/accelerometers>

P3D. (n.d.). Retrieved from processing.org: <https://processing.org/tutorials/p3d/>

Rendering. (2021, March 10). Retrieved from Wikipedia:

[https://en.wikipedia.org/wiki/Rendering_\(computer_graphics\)#:~:text=Rendering%20or%20image%20synthesis%20is,means%20of%20a%20computer%20program.&text=The%20data%20contained%20in%20the,or%20raster%20graphics%20image%20file](https://en.wikipedia.org/wiki/Rendering_(computer_graphics)#:~:text=Rendering%20or%20image%20synthesis%20is,means%20of%20a%20computer%20program.&text=The%20data%20contained%20in%20the,or%20raster%20graphics%20image%20file).