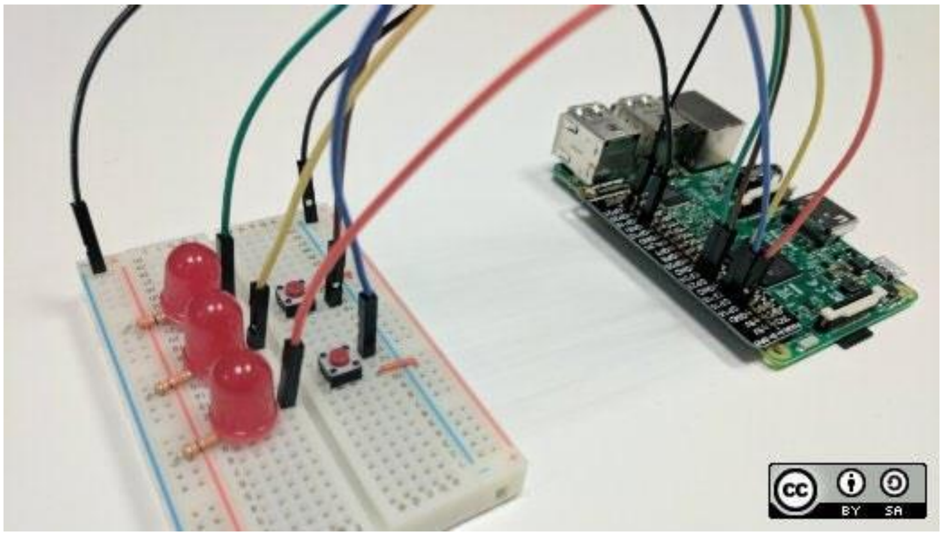


PHYSICAL COMPUTING PORTFOLIO

Student's Full Name: Maxene Januelle Gaspay Salon

BSU ID: 581986

Tutor: Ms. Arshiya Subhani

Department: Creative Computing

Education: BathSpa University Academic Center, RAK, UAE

Sway link: PHYSICAL COMPUTING PORTFOLIO

Google Drive link: Google Drive

YouTube links

Experiment 1- Buzzer: https://youtu.be/crzbtKIKqw0

Experiment 2- LED: https://youtu.be/Tl7Z9FvvB9o

Experiment 3- Radar: https://youtu.be/TF8SzQUtn4Q

Experiment 4- Real-Time Clock: https://youtu.be/DumLCmTf6Ng

Experiment 5- LCD Display: https://youtu.be/q6rgbK5pnnI

Experiment 6- RFID: https://youtu.be/9mQcN8goW9w

Final Project- Mechanical Arm: https://www.youtube.com/watch?v=yy9pJxxkZLQ

INTRODUCTION TO PHYSICAL COMPUTING

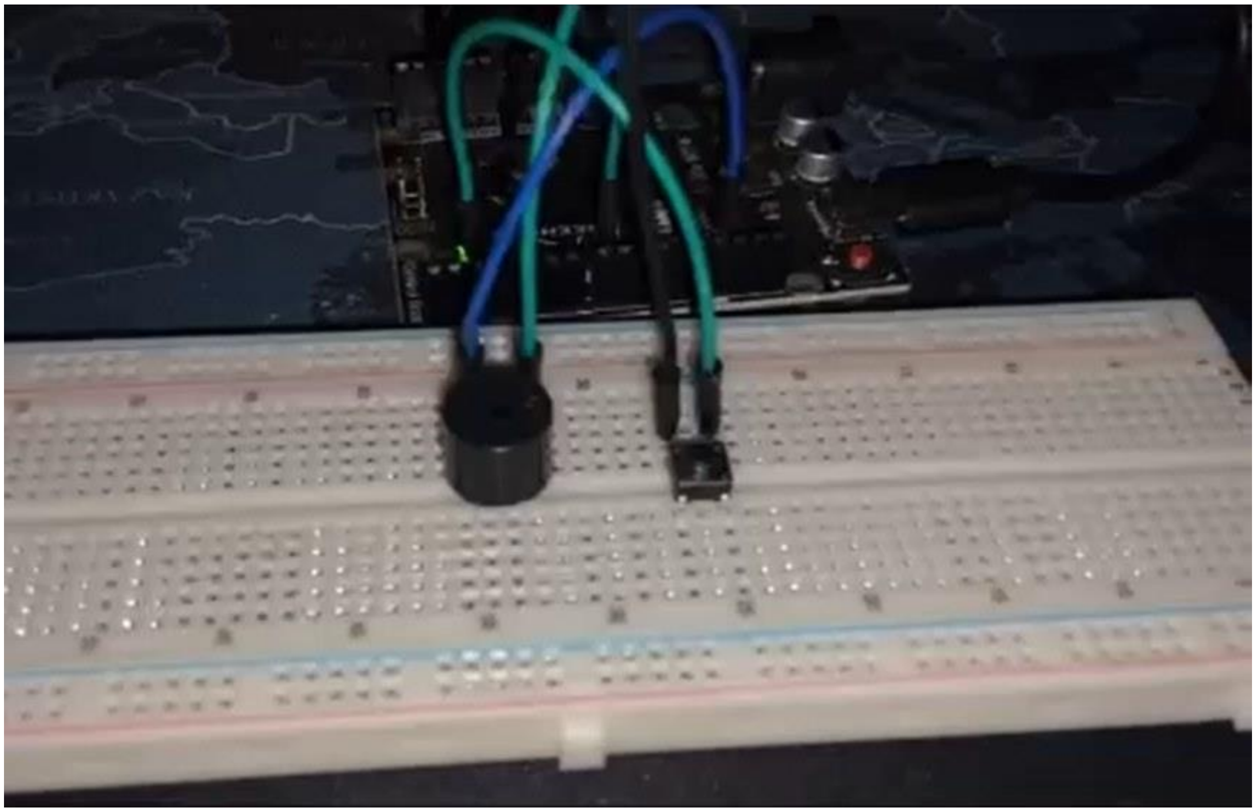
GENERAL INFORMATION

Physical Computing is an exciting field that merges hardware and software into interactive systems that can sense and respond to the real world. For me, it’s where creativity meets engineering. With microcontrollers, sensors, and a bit of code, we can build machines that act as our assistants, smart, reactive, and functional. This is the backbone of the tech we rely on daily: from smartphones and smartwatches to automated doors and digital displays. What fascinates me most is how physical computing doesn’t just power innovation, it *is* innovation. It bridges the gap between imagination and reality, allowing us to build things that improve convenience, solve real-world problems, or even just make life a bit more enjoyable. It’s not just about functionality, it’s about taking pride in what we create, knowing that even the smallest project could spark something greater.

PREMISE

Since this project focuses on innovation and invention, my portfolio includes a set of hands-on mini-experiments that we, as students, were tasked to build ourselves. These include up to six devices: the buzzer, LED, LCD, RFID, Motion Radar, and a real-time clock. Building each of these gave me the chance to understand not just the theory, but the actual process, from wiring to writing code, and seeing everything come to life. These small machines may seem simple, but they represent the beginning of something much bigger. They teach us how to think like makers, problem-solvers, and future innovators. Every idea we build now has the potential to evolve into something meaningful for the world. As technology continues to grow, we’re not just watching it happen, we’re learning to be part of that growth. The future is being shaped, and we're building its foundation, one wire at a time.

BUZZER



The application of basic electronic components, such as buzzers, provides an excellent entry point for students to see hardware programming in action. Buzzers are small sound-making devices, mainly used for producing an audible alert or providing feedback in systems like alarms, notifications, or timers. Joined together with an Arduino microcontroller, the buzzer becomes an instrument with which one learns to first interface electronically, signal control, and microcontroller programming.

Materials required for the project: Arduino Uno (or any compatible board), **Piezo Buzzer** (either active or passive), **Breadboard, Jumper Wires, USB Cable** (to upload the code to Arduino), **Arduino IDE** (installed on a PC or laptop). These are cheap and very much available; hence they are the ideal choice for either classroom or personal projects.

Circuit Setup

The circuit has an easy setup. To begin, **insert the buzzer into the breadboard**. This buzzer usually has two pins: one positive (+) and one negative (–). **The positive pin is connected to**

digital pin 8 **on the Arduino by using a jumper wire**, while the **negative pin is connected to GND** on the Arduino. One must ensure the connection of the red and black is appropriate to avoid damage and malfunctioning. With an active buzzer, the Arduino can turn it on and off. A passive buzzer allows you to control sound frequency: thus, custom tones or melodies can be played.

Click here to download the code

Click here to view the video

LCD DISPLAY

In today's world of electronics and embedded systems, making connections simpler and cutting down on pin usage is crucial for creating efficient and scalable projects. One effective way to do this is by utilizing the I2C (Inter-Integrated Circuit) protocol, which enables multiple devices to communicate using just two wires. When you combine it with an LCD (Liquid Crystal Display), the I2C module allows you to display information with minimal wiring and setup. This project will guide you through connecting and programming an I2C-based 16x2 LCD display with an Arduino, making it a perfect starting point for learning about serial communication and display interfacing.

Materials Required

To complete this project, you will need the following components: **Arduino Uno** (or any compatible board), **16x2 LCD Display with I2C Module** (or a regular 16x2 LCD with an I2C backpack), **Jumper Wires, Breadboard (optional), USB Cable** (for uploading code), **Arduino IDE** (installed on your PC). This setup is simpler than the standard LCD configuration, as the I2C module reduces the required connections from 12+ pins to just four.

Circuit Setup

The I2C module has four pins labeled:

- **GND** > Connect to Arduino **GND**
- **VCC** > Connect to Arduino **5V**
- **SDA** > Connect to Arduino **A4** (on Uno)
- **SCL** > Connect to Arduino **A5** (on Uno)

That's it. This minimal wiring is one of the biggest advantages of using I2C, especially when working with multiple components or sensors.

Click here to download the code

Click here to view the video

MOTION RADAR

Motion detection plays a vital role in a variety of modern technologies, including security systems, robotics, and automation. A widely used method for real-time motion detection involves ultrasonic sensors. These sensors send out sound waves and track how long it takes for the echo to bounce back after hitting an object. When paired with an Arduino microcontroller, they can function like a basic radar system, identifying both the presence and distance of moving objects within a certain range. This project is a great way for students to learn about ultrasonic sensing, how to interface with microcontrollers, and the ins and outs of real-time data processing.

Materials Needed

To complete this radar motion detection system, the following materials are required: **Arduino Uno** (or any compatible board), **HC-SR04 Ultrasonic Sensor, SG90 Servo Motor** (to rotate the sensor), **Breadboard, Jumper Wires, USB Cable** (for uploading code and serial communication), **Arduino IDE** (for writing and uploading the microcontroller code), **Processing IDE** (for data visualization on the computer). These components are all easily accessible and provide a cost-effective platform for developing an interactive motion detection system.

Circuit Setup

The circuit involves connecting both the ultrasonic sensor and a servo motor to the Arduino. The **HC-SR04 sensor** is mounted on the **servo**, which rotates it to scan the surroundings:

- **Ultrasonic Sensor Connections**:
    - **VCC** > **5V** on Arduino
    - **GND** > **GND** on Arduino
    - **Trig** > Digital **Pin 9**
    - **Echo** > Digital **Pin 10**
- **Servo Motor Connections**:
    - **VCC** (red) > **5V**
    - **GND** (brown/black) > **GND**
    - **Signal** (orange/yellow) > Digital **Pin 11**

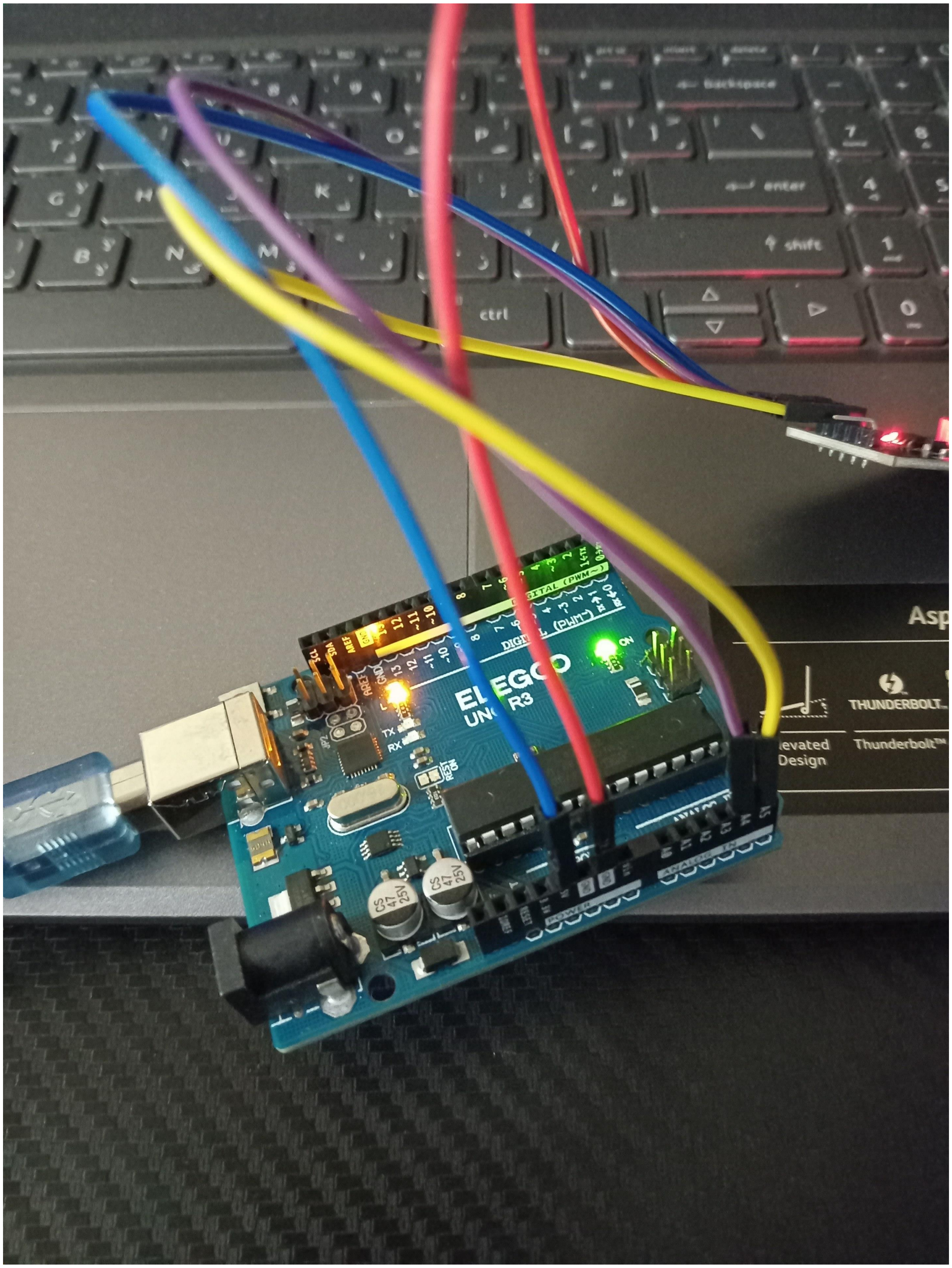
This setup allows the ultrasonic sensor to pivot in a radial arc, capturing distance data at each angle.

Click here to download the code (Arduino)

Click here to download the code (Processing)

Click here to view the video

REAL-TIME CLOCK



In both real-world applications and digital systems, time is an essential element. Timekeeping enables embedded devices, such as data loggers and alarm clocks, to plan events, automate procedures, and keep time. An external Real-Time Clock (RTC) module is necessary for any project that requires precise timekeeping—even when the power is turned off—because Arduino microcontrollers lack built-in persistent clocks. In order to maintain and display real-time data, this project investigates the use of a common RTC module, such as the DS1307 or DS3231, connected to an Arduino.

Materials Needed

To create a working RTC project using Arduino, the following components are typically used: **Arduino Uno** or compatible microcontroller board, **RTC Module (DS1307 or DS3231), CR2032 Coin Cell Battery** (to power the RTC independently), **Jumper Wires, Breadboard** (optional), **USB Cable** (for programming and serial communication), **Arduino IDE** (installed on a computer for writing and uploading code). Some projects may also include an **LCD display** or **OLED screen** to show the time, but for basic testing, a computer's serial monitor is sufficient.

Circuit Setup

The RTC module uses the **I2C communication protocol**, which only requires two signal lines:

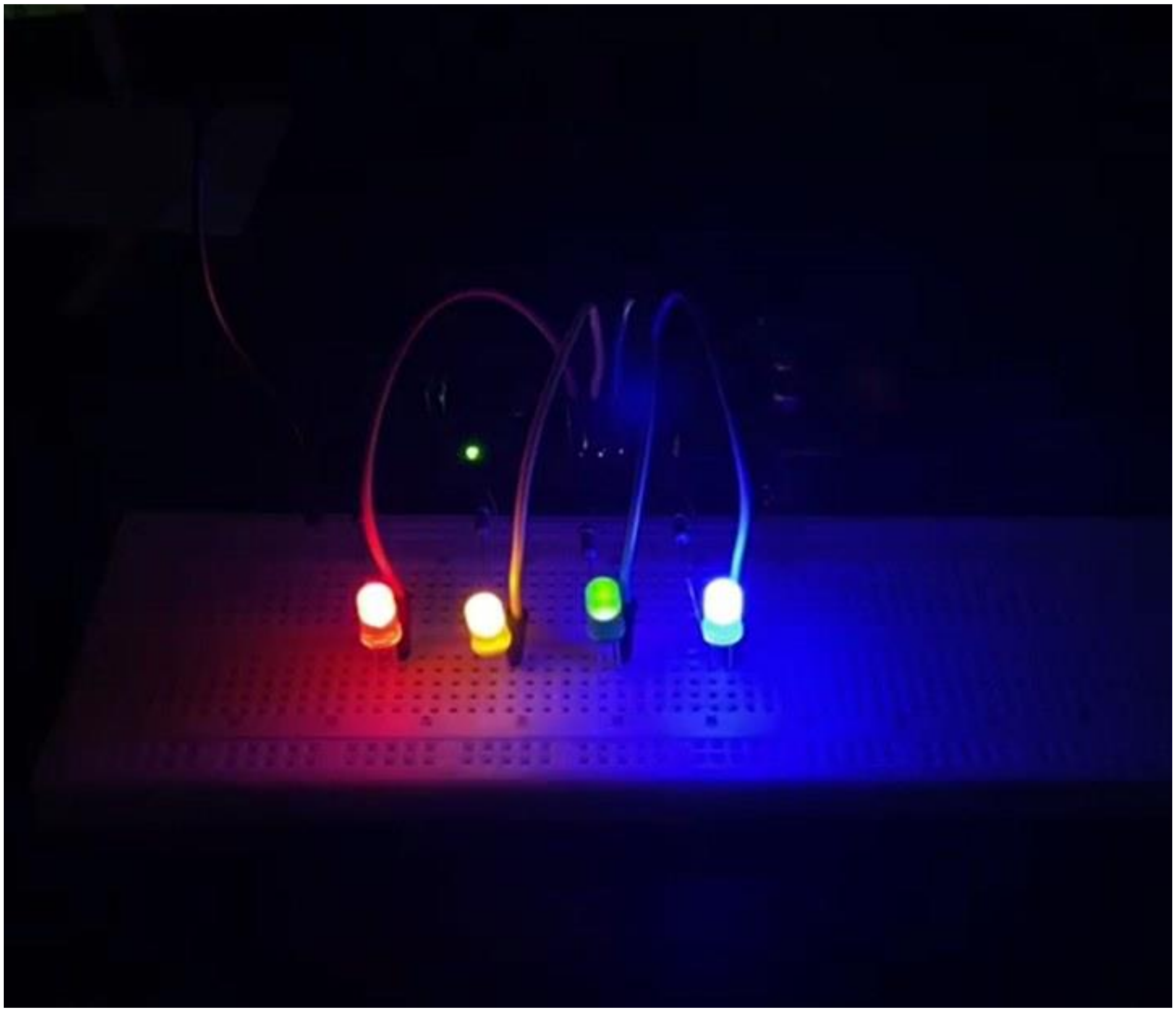
- **VCC** → Connect to Arduino **5V**
- **GND** → Connect to Arduino **GND**
- **SDA (Data Line)** → Connect to **A4** on Arduino Uno
- **SCL (Clock Line)** → Connect to **A5** on Arduino Uno

The coin cell battery is inserted into the RTC module to keep the time running even when the Arduino is unplugged. This ensures long-term accuracy and is one of the key advantages of using an RTC.

Click here to download the code

Click here to view the video

LED INDICATOR



Light-emitting diodes (LEDs) are among the most fundamental yet crucial parts of the electronics and physical computing industries. Despite their simplicity, LEDs are essential for giving users real-time feedback, whether it be for circuit debugging, status update notifications, or signaling errors and activity in embedded systems. An Arduino microcontroller can be used to precisely control an LED indicator that reacts to external inputs like sensors or code-based commands. This project demonstrates how an LED can be utilized as an output device in embedded applications, offering both practicality and fundamental electronics knowledge.

Materials Needed

To build an LED indicator system using Arduino, you only need a few components, most of which are inexpensive and widely available: **Arduino Uno** (or any compatible board), **LED** (any color, preferably red, green, or blue for clarity), **220Ω Resistor** (to prevent damage to the LED), **Breadboard** (optional, for easy prototyping), **Jumper Wires, USB Cable** (to upload code and power the Arduino), **Arduino IDE** (installed on a PC or laptop). Optional components like push buttons or sensors can be added to extend the functionality, but the base project requires just an LED and resistor.

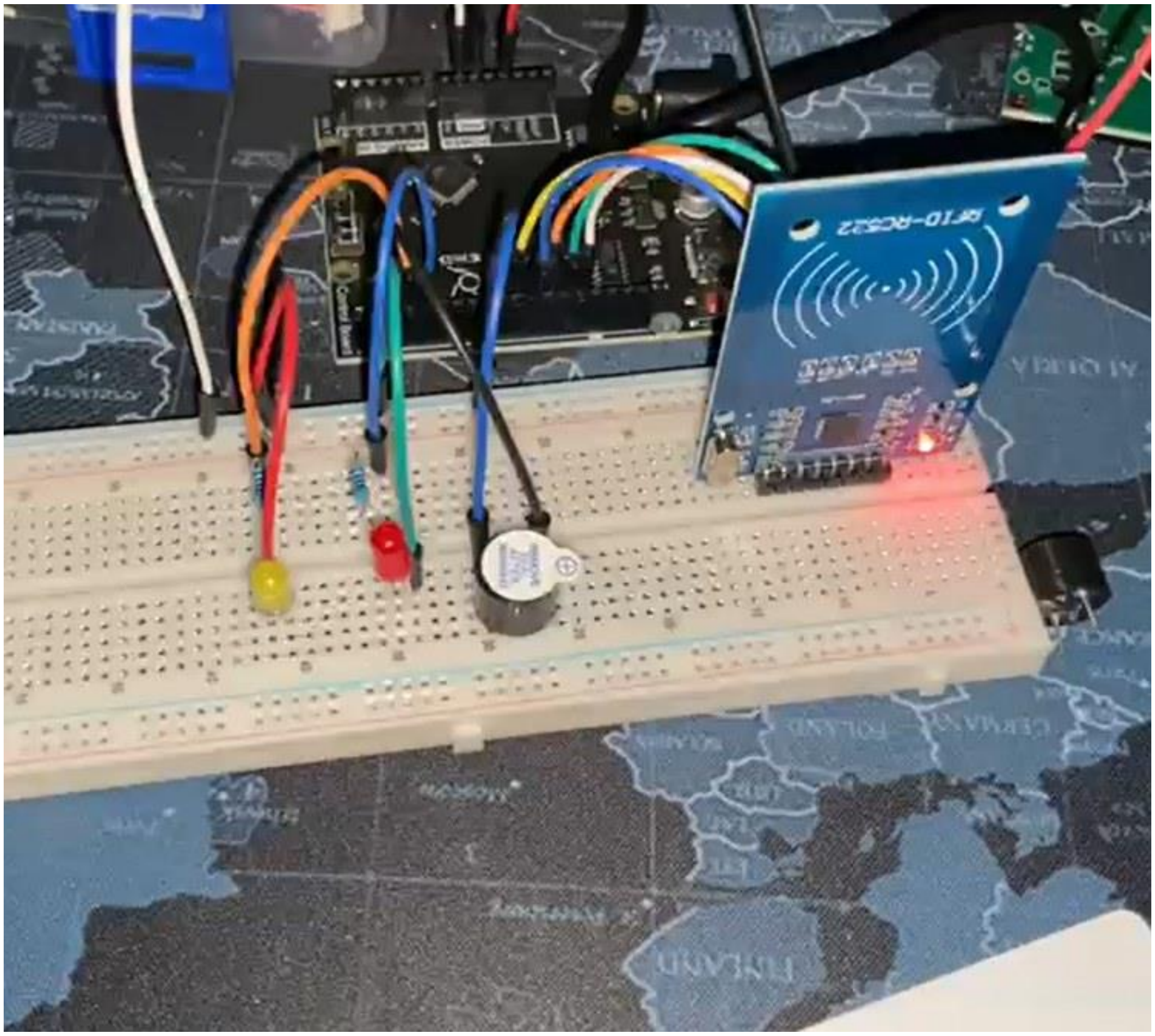
Circuit Setup

The basic circuit involves connecting one leg of the LED (the **anode**, or longer leg) to a **digital output pin** on the Arduino — typically pin 13, which often has a built-in LED as well. The other leg (the **cathode**) connects to one side of the resistor, and the other end of the resistor is connected to the **GND (ground)** pin of the Arduino. This setup ensures that the current flowing through the LED is limited by the resistor, protecting the LED from being overloaded and burned out. The resistor value (usually between 220Ω to 330Ω) is chosen based on the LED's voltage drop and the output voltage of the Arduino pin (typically 5V).

Click here to download the code

Click here to view the video

RFID



RFID (Radio-Frequency Identification) technology has grown ever more important as modern society advances toward automation and improved security systems. RFID lets devices read data from a tag wirelessly, without physical contact, so enabling access control systems, inventory management, and contactless payment systems. Building an RFID system with Arduino gives students practical knowledge with digital communication, data validation, and security protocols in the framework of physical computing and microcontroller-based design. < Those studying the principles of authentication systems will especially find this project highly pertinent.

Materials Needed

To implement a simple RFID-based access system using Arduino, the following components are required: **Arduino Uno** (or compatible board), **RFID Module (MFRC522), RFID Tags or Cards (13.56 MHz), Jumper Wires, Breadboard** (optional), **USB Cable** (for Arduino-to-PC connection), **Arduino IDE** (software to upload code and read serial data). Optional elements might include an **LCD display** to show user status or an **LED and buzzer** for access indicators.

Circuit Setup

The MFRC522 RFID module communicates with Arduino via the **SPI (Serial Peripheral Interface)** protocol. It typically includes 8 pins, though only 7 are usually used in basic configurations:

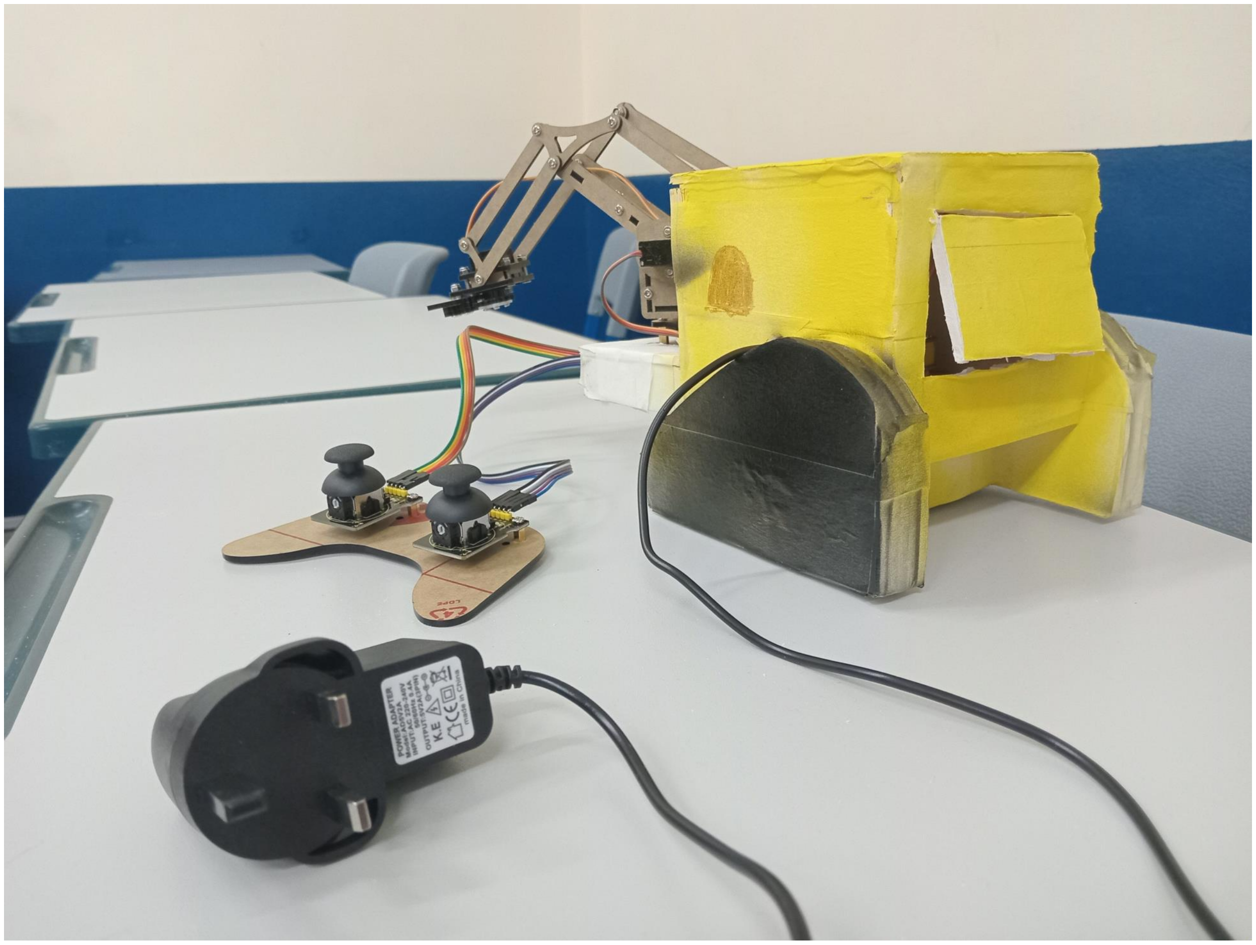
- **VCC** → Connect to Arduino **3.3V** (not 5V, to avoid damaging the module)
- **GND** → Connect to Arduino **Ground**
- **SDA (SS)** → Connect to **Pin 10**
- **SCK** → Connect to **Pin 13**
- **MOSI** → Connect to **Pin 11**
- **MISO** → Connect to **Pin 12**
- **RST** → Connect to **Pin 9**

These connections enable the Arduino to read data sent by the RFID module when a tag is brought near its antenna.

Click here to download the code

Click here to view the video

MECHANICAL ARM (TEAM 5 PROJECT)



INTRODUCTION | GROUP 5

Robotic arms are widely used in industrial automation, medical applications, and educational robotics for performing precise and repeatable tasks. In this project, we used an Arduino microcontroller to build and operate a mechanical arm kit from Keyestudio, allowing us to investigate the foundations of robotics. We can learn about servo motor control, movement coordination, and the fundamentals of mechanical design with the Keyestudio arm, which offers a simplified but useful model of a robotic manipulator. In order to replicate the flexibility of a human arm, the arm is made up of several joints that are each driven by a servo motor. We were able to regulate the location and order of movements by setting up the Arduino to communicate with each servo. Through this project, we learned about important mechatronics concepts like serial input control, kinematics, and PWM (pulse-width modulation). The mechanical arm provided a practical setting for learning how computer commands result in motion. It is a first step toward creating increasingly sophisticated robotics systems for automation, artificial intelligence, and manufacturing.

MATERIALS, PROCESS, & PREPARATIONS

Materials Used

Electronics & Components:

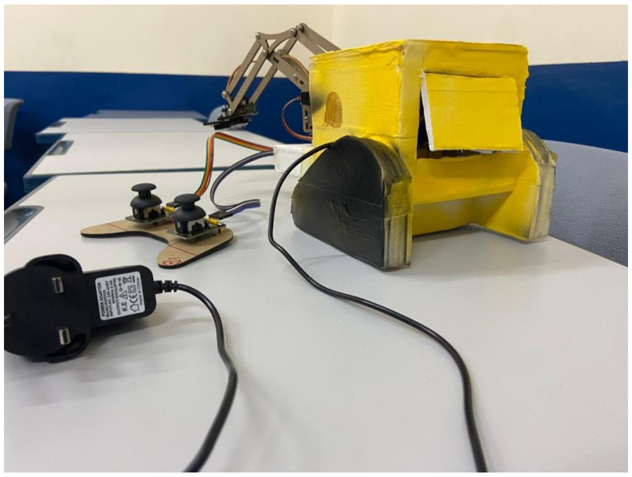
- Arduino Uno board
- Keyestudio mechanical arm kit (includes 4x servo motors and plastic frame components)
- Servo motor driver board (included in the kit)
- Jumper wires (male-to-female)
- USB cable (for uploading Arduino code)

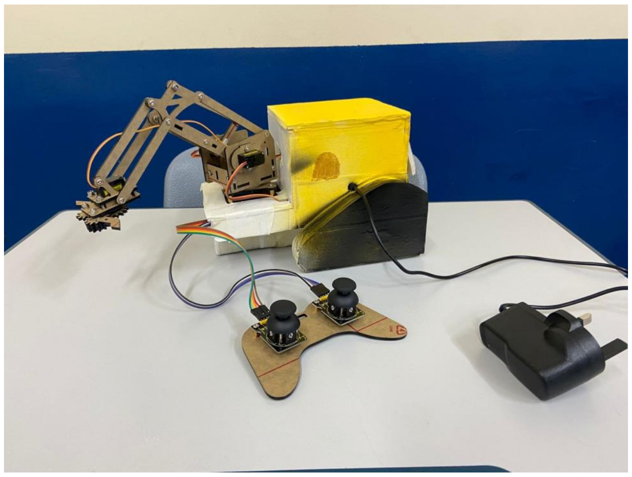
Structural Materials:

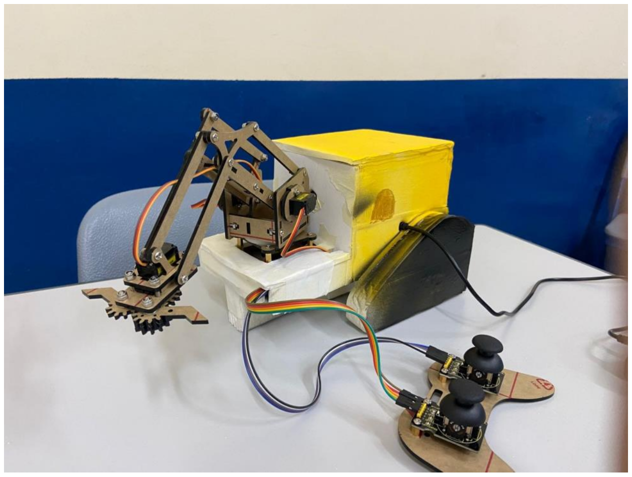
- Cardboard (for added support, custom extensions, or base stability)
- Duct tape (to reinforce weak joints, secure parts, and stabilize the base)
- Double-sided tape or glue (optional, for attaching cardboard components)

Tools & Software:

- Screwdriver (for assembling frame and servo brackets)
- Scissors/cutter (for shaping cardboard)
- Arduino IDE (for programming and uploading code)







[Click here to download the code]()

[Assembling and Coding Process with the Members]()

[Click here to view the video]()

STEPS

In order to build the claw, arm segments, and base of the mechanical arm, Group 5's task started with locating materials like wood, acrylic, or cardboard—or, alternatively, choosing to use 3D-printed components. In order to ensure that the servo motors in each joint met the mechanical requirements of their locations, we carefully chose and installed them. Even though the wiring process was difficult, we carefully followed the directions to make sure everything was connected and working properly. The base was then equipped with a power supply to supply the energy required to operate the arm. After that, we used the Arduino IDE to program the arm, defining motor pins and writing movement sequences by integrating the servo library. We made certain that the code was appropriately organized, stored, and prepared for testing. We improved the system through several test runs. When mistakes happened, we worked together to identify the problems, make the necessary adjustments, and upload the updated code again until the arm was functioning as it should have.

CRITICAL REFLECTION

Building and programming the mechanical arm was a difficult but enlightening experience that demonstrated the value of teamwork in group projects in addition to technical knowledge. We discovered a number of personal takeaways during the design, assembly, and coding phases, ranging from enhanced problem-solving abilities to a better comprehension of group dynamics.

Tayyeb, Basil, and I were mostly responsible for the first stages, which included wiring the servo motors and assembling the mechanical arm's physical parts. Precise coordination, perseverance, and a mutual dedication to closely adhere to the hardware documentation were necessary for this phase. When we started to see movement in our prototype, it was finally worthwhile to align the servos to the proper positions and make sure the connections were stable. We also had to deal with unforeseen hardware-related issues during this phase, such as loose or misaligned connections that needed numerous adjustments and double-checks.

Our team had to switch from manual assembly to logical command and motion sequence structuring when we entered the coding phase. We created movement routines for the various arm joints using the Arduino Servo library, allocating the proper pins for each motor. Several testing cycles were necessary for this stage, and we ran into issues with synchronization and power constraints. Again, Tayyeb, Basil, and I worked together to debug these issues and ensure the system functioned smoothly.

As the project came to a close, the emphasis shifted to improving the mechanical arm's overall appearance and structural stability, streamlining user interactions, and optimizing the code. Christian Patrick and I were in charge of this section. We worked together to improve the physical build's appearance, make sure the code was clear and well-commented, and refine the movement logic. To achieve a cleaner look, this involved moving parts around, covering wiring, and strengthening certain areas with cardboard and duct tape.

Even though the project was technically successful, team participation was not without its difficulties. Regretfully, Saif and Umer, two group members, were mostly absent during the procedure. They made no significant contributions to the programming or build phases. Due to the imbalance in workload caused by their lack of participation, the rest of us were under more pressure to finish the project on schedule. This brought to light a critical aspect of group work: accountability within a team can be challenging to enforce, and even in real-world projects, not all members may participate equally.

Nevertheless, those of us who actively participated learned a lot about not only how to construct a working mechanical arm but also how to work well with others under duress. Our knowledge of servo control, hardware-software integration, and the perseverance needed for prototyping has increased as a result of the project. Above all, it taught us that initiative and communication are just as important to the success of any group project as technical proficiency.