CSE 2006

MICROPROCESSOR AND INTERFACING

Digital Assignment – 1

E2 | SJT215

FALL SEMESTER 2021-22

by

SHARADINDU ADHIKARI

19BCE2105

Q. **Write an Introduction to Arduino Boards, which covers the following points:**

- What makes up an Arduino Board?
- What can the Arduino do?
- Why use the Arduino?
- Arduino Sensors and Shields.
- Explain its one Application with Case Study.

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. We can tell our board what to do by sending a set of instructions to the microcontroller on the board. To do so we use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

What makes up an Arduino Board?

The design has changed through the years, and some variations include other parts as well. But on a basic board, we're likely to find the following pieces:

- A number of pins, which are used to connect with various components we might want to use with Arduino. These pins come in two varieties:
    - Digital pins, which can read and write a single state, on or off. Most Arduinos have 14 digital I/O pins.
    - Analog pins, which can read a range of values, and are useful for more fine-grained control. Most Arduinos have six of these analog pins.

  These pins are arranged in a specific pattern, so that if we buy an add-on board designed to fit into them, typically called a "shield," it should fit into most Arduino-compatible devices easily.

- A power connector, which provides power to both the device itself, and provides a low voltage which can power connected components like LEDs and various sensors, provided their power needs are reasonably low. The power connector can connect to either an AC adapter or a small battery.

- A microcontroller, the primary chip, which allows us to program the Arduino in order for it to be able to execute commands and make decisions based on various input. The exact chip varies depending on what type of Arduino we buy, but they are generally Atmel controllers, usually a ATmega8, ATmega168, ATmega328, ATmega1280, or ATmega2560. The differences between these chips are subtle, but the biggest difference a beginner will notice is the different amounts of onboard memory.

- A serial connector, which on most newer boards is implemented through a standard USB port. This connector allows us to communicate to the board from our computer, as well as load new programs onto the device. Often times Arduinos can also be powered through the USB port, removing the need for a separate power connection.

- A variety of other small components, like an oscillator and/or a voltage regulator, which provide important capabilities to the board, although we typically don't interact with these directly.

Arduino RS232[38] (male pins)

Arduino Diecimila[39]

Arduino Duemilanove[40] (rev 2009b)

Arduino Uno R2[41][42]

Arduino Uno SMD R3[43]

Arduino Leonardo[44]

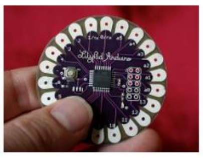
Arduino micro (AtMega 32U4)

Arduino pro micro (AtMega32U4)

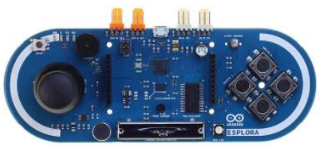
Arduino Pro[45] (No USB)

Arduino Mega[46]

Arduino Nano[47] (DIP-30 footprint)


Arduino LilyPad 00[48] (rev 2007) (No USB)

Arduino Robot[49]


Arduino Esplora[50]

Arduino Ethernet[51] (AVR + W5100)

Arduino Yún[52] (AVR + AR9331)

Arduino Due[53] (ARM Cortex-M3 core)

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. 17 versions of the Arduino hardware have been commercially produced.

What can the Arduino do?

It depends on whether we're only talking about the bare Arduino, or Arduinos with shields or other augmented bits attached. One great thing about Arduinos is that it's relatively easy to design shields that add capabilities to the Arduino. Without anything attached, the Arduino is pretty useless. It has no way to interact with the outside world other than USB serial.

Once we've got peripherals added to our Arduino, we communicate wirelessly, read sensors, trigger motors, audio outputs, light shows, etc. The sky's the limit. I especially like being able to put an entire microcontroller development environment, including programmer and an assortment of components (jumper wires, resistors, etc) in my laptop bag and go off to a coffee shop to hack on it.

Some of the really cool stuff Arduino can do:

- Build a 3D printer.
  This is possibly the greatest thing we can make out of an Arduino because adding a 3D printer to our toolset will enable us to build more complex projects that include custom enclosures and interfaces. Best of all, there are Arduino-based 3D printer builds that can suit any budget.

- Make an AR-Augmented Laser Cutter.
  If you've ever seen a modern laser cutter in action, you were probably amazed at the precision and speed with which it can etch or cut shapes into a variety of materials. With Arduino, we can make one of our very own – with a unique twist. We can build an AR-augmented Laser Cutter that will let us create detailed cutouts with no complex and precise design work beforehand.

- Build Your Own Wall-E.
  Whenever anyone thinks of building their own electronics, the first image they're likely to conjure up is of some type of homebrew robot. Well, an Arduino can't power a complex robot, it can help us to build a working replica of everyone's favourite animated robot – Wall-E. With an Arduino for a brain and Lego for his body, we'll have our very own Wall-E motoring around our home, avoiding obstacles and delighting our guests.

- DIY Fingerprint Scanning Garage Door Opener.
  If you've ever come home to a locked door, only to realize you don't have your keys, the feeling is mutual. This Arduino project is a fingerprint scanning garage door opener, so we'll never be locked out in the cold again. All it requires is an Arduino board, a handful of electronic components, and a 3D printer case (which we can make ourselves if we already completed the first project on this list).

- Arduino Mega Chess.
  We'll need a TFT LCD touch screen display and an Arduino Mega 2560 board as the primary materials. If we have a 3D printer, we can create a pretty case for it and make changes accordingly.

- Robot Arm with Controller.
  If we want to do something with the help of a robot and still have manual control over it, the robot arm with a controller is one of the most useful Arduino projects. It uses the Arduino UNO board. We will have a robot arm for which we can make a case using the 3D printer to enhance its usage and we can use it for a variety of use-cases.

- Audio Spectrum Visualizer.
  For this, we will need an Arduino Nano R3 and an LED display as primary materials to get started with. We can tweak the display as required. We can connect it with our headphone output or simply a line-out amplifier.

Having said all that, it might be useful to also ask "**What can an Arduino *not* do?**". Arduino has very limited memory and I/O compared to modern computers. So, some of the things the Arduino cannot do or cannot do easily are things that require a lot of memory or access to complex peripherals, like:

- video recording, processing & output
- high-fidelity audio recording & processing
- act as USB host for USB devices like flash drives, disk drives, cameras, keyboards, etc.

So, we can't easily make a video game system that hooks to our TV with an Arduino. That doesn't mean people haven't done it, but that level of hacking is in the realm of deep voodoo, and the results still end up looking like a 1980s videogame.

Often, we see Arduino boards hooked up as peripherals to a larger computer. The computer does the A/V and the Arduino handles all the other physical world interfacing. Another common use is the fully embedded system where a more mundane device is made "smart" with a hidden Arduino. ("Your coffee table now knows when you set cups on it and buzzes you if you don't use a coaster") This is where Arduino seems happiest.

Why use the Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. Teachers and students use it to build low-cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step-by-step instructions of a kit, or sharing ideas online with other members of the Arduino community.

Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50.

- Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

- Open source and extensible software - The Arduino software is published as opensource tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, we can add AVR-C code directly into our Arduino programs if we want to.

- Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Arduino Sensors and Shields.

Sensors:

Arduino sensors are defined as a module that detect changes in the environment. The sensors transfer those changes to the electronic devices in the form of a signal. It is made up of Single Crystal Silicon, and is considered as a widely used semiconductor material. It has superior mechanical stability, machinability, etc. It can also combine electronics and sensing elements on the same substrate.

The data signal runs from the sensor to the output pins of the Arduino. The data is further recorded by the Arduino.

Some of the types of sensors in Arduino are listed below:

- Light sensor: It is used to control the light. It is used with LDR (Light Dependent Resistor) in Arduino.
- Ultrasonic sensor: It is used to determine the distance of the object using SONAR.
- Temperature sensor: It is used to detect the temperature around it.
- Knock Sensor: It is used to pick the vibrations of the knocking. It is a common category of a vibration sensor.
- Object Detection Sensor: It is used to detect the object by emitting infrared radiations, which are reflected or bounced back by that object.
- Tracking Sensor: It allows the robots to follow a particular path specified by sensing the marking or lines on the surface.
- Metal Touch Sensor: It is suitable for detecting the human touch.
- Water Level Sensor: It is used to measure the water or the depth of the water level. It is also used to detect leaks in containers.
- Vibration Sensor: The vibration sensor is used to measure the vibrations.
- Air Pressure sensor: It is commonly related to meteorology, biomedical fields.
- Pulse Sensor: It is used to measure the pulse rate.
- Capacitive soil moisture sensor: It is used to measure the moisture level of the soil.

- Microphone sensor: This sensor in Arduino is used to detect the sound. The analog and digital are the two outputs of this module. The digital output sends the high signal when the intensity of sound reaches a certain threshold. We can adjust the sensitivity of a module with the help of a potentiometer.
- Humidity sensor: It is used to monitor weather conditions.
- Motion sensor: It detects the movement and occupancy from the human body with the help of Infrared radiation.
- Vibration sensor: It is used to detect the vibrations.
- Sound sensor: It is suitable to detect the sound of the environment.
- Pressure Sensor: It is used to measure the pressure. The sensor in Arduino measures the pressure and displays it on the small LCD screen.
- Magnetic field sensor: It measures the magnetic field strength and produces a varying voltage as the output in Arduino.

Shields:

An Arduino Shield is a pre-made board that stacks on top of the Arduino board using the headers and pins. Arduino Shields are extremely convenient when we are beginning to prototype an idea; they take the guess work out of interfacing with sensors and input devices by designing the circuit for us.

A brief list of Arduino Shields:

- Prototype Shield: Perhaps the simplest of Arduino Shields is the Prototype Shield. It comes with a prototyping area, on which, we can solder the components, if necessary. If we do not want to solder, then don't worry. The shield also comes with a 170 Pin Mini breadboard, which can be attached on the prototyping area with the help of double-sided tape.

- IO Expansion Shield: As the name suggests, an IO Expansion Shield allows us to connect several Analog and Digital IO devices to the Arduino without breadboard and soldering. There are headers for connecting 3-pin and 4-pin sensors.

- Multifunction Shield: It contains 4 LEDs, 3 Push Buttons, a 10 kΩ Potentiometer, a Piezo Buzzer as the basic IO devices. There is also a serial interface header for connecting Serial Modules like Bluetooth, Voice Recognition, Wireless, Voice, etc.

- LCD Shield: One of the popular Arduino Shields is the LCD Shield. It is built around the famous 1602 Character LCD (16×2 LCD Module). It contains a 16×2 LCD Display with White characters and Blue backlight. The shield also contains 6 Push Buttons of which 1 is the Reset button and the other 5 are for user application like LEFT, RIGHT, UP, DOWN and SELECT.

- Joystick Shield: Controlling Robots and RC Cars with a Joystick is a fun little project on its own. A Joystick Module is a tricky one as it is not breadboard friendly. So, using a Joystick Shield on top of our Arduino Board changes the "game" completely (pun intended). It contains a 2-Axis Joystick Module, 6 Push Buttons, Serial Interface Connector, I2C Interface Connector and a dedicated nRF25L01 Module Connector.

- Relay Shield: A Relay Shield is an Arduino Expansion Board consisting of 4 Mechanical Relay Modules with four dedicated terminal connectors for each relay. It also consists of four LEDs to indicate the state of the Relay (NO or NC).

- 4×4 Keypad Shield: A 4×4 Keypad Matrix consists of 16 Push Buttons arranges in a matrix of 4 Rows and 4 Columns. Each button can be mapped to a character or a value. To simplify the interface, we can use the 4×4 Keypad Shield.

- Capacitive Touchpad Shield: To incorporate Touchpad in our Arduino Project, we can use the Capacitive Touch Pad Shield. It consists of 9 Capacitive Touch Pads interface through MPR121 IC, Proximity Capacitive Touch Sensor Controller.

- GSM/GPRS Shield: Using a GSM/GPRS Shield, we can connect our Arduino Board to a GSM Network. The GSM/GPRS Shield allows us to make / answer calls, send / receive messages (SMS), connect to internet through GPRS.

- Bluetooth Shield: HC-05 Bluetooth Module is a very popular communication modules for Arduino. There are Arduino Shields with Bluetooth Modules on then to enable Bluetooth Communication over serial interface.

- Ethernet Shield: Another popular shield in the Arduino Community is the Ethernet Shield. This particular Ethernet Shield is based on W5100 Ethernet Controller from Wiznet. Connect an ethernet cable to the RJ-45 Jack and we can control Arduino from the Internet.

- Wi-Fi Shield: The impact of ESP8266 on DIY Community is immeasurable. It is a small module with built-in Wi-Fi for wireless connectivity of Arduino. The combination of Arduino and ESP8266 is a major contributor to the DIY IoT Projects. There are Wi-Fi shields for Arduino to easily integrate ESP2666 (or any other Wi-Fi Controller) to an Arduino Board.

- MP3 Player Shield: The VS1053 MP3 Decoder based MP3 Shields for are Arduino are a great way to add Music touch to our DIY Project. The VS1053 MP3 Decoder IC is an decode several audio formats like Ogg Vorbis, MP3, AAC, WMA, MIDI.

- GPS Shield: A GPS Shield with Data Logger consists of a GPS Receiver Module and a microSD Card Slot on-board. The GPS Module interfaces with Arduino over serial communication while the microSD card is connected to the SPI Pins.

- MQ2 Smoke Sensor Shield: The MQ2 Gas Sensor is a very useful module to implement safety related applications. An MQ2 Smoke Sensor Shield will be perfect for our Arduino Board to detect Smoke, LPG, Carbon Monoxide and other toxic gases.

- Camera Shield: Interfacing camera with a Microcontroller board like Arduino is always a challenging task as the camera is a memory intensive module. But the Camera Shield for Arduino by ArduCam simplifies this task with a easy to use camera and simple user interface.

- Energy Shield: If we want to add power backup to our Arduino Project, then Energy Shield is the way to go. This shield is based on LiPo Battery power shield, which will charge the battery if external power is available but switches to battery power in case the external power is disconnected.

- FM Radio Shield: Digital FM Radio Shield for Arduino enable us to listen to FM stations using our Arduino Board. We can digitally control the stations, receive Radio data system (RDS) information like artist song and read the strength of the signal using this shield. The on-board DSP Signal Conditioning system provides us with a clear audio over the headphones. There are also buttons to change volume and stations.

Explain one of its application with Case Study.

For this purpose, I'm considering using Arduino as a Case Study for Automation Prototyping.

The Need:

In extreme macro photography, you are shooting very small subject at very close focusing distances. This has an inherent limitation that the area of the subject that is in focus is extremely small - sometimes in few microns, like the thickness of a human hair. So, to overcome this what is done is called focus stacking. We take multiple images and in each image the focus plane is moved by moving the camera. Then all these images are processed in a software which merges the areas in focus to come to final image which will have everything in focus.

This usually uses a focus rail where the camera is mounted and each time we rotate a knob to move the camera to the next plane to be focused at and then take a photo. The objective of the project was to automate this sequence, move - shoot - move - shoot...... till the end point is reached.

An example of a single image is as shown below.

Objectives:

1. to be able to specify for each photo-shoot, the starting point, ending point and step between photos.
2. to be able to move the camera precisely in equal steps from beginning to end of the sequence.
3. to be able to trigger the camera at the end of each step.

Execution:

To execute the project, I had multiple interfaces to be handled.

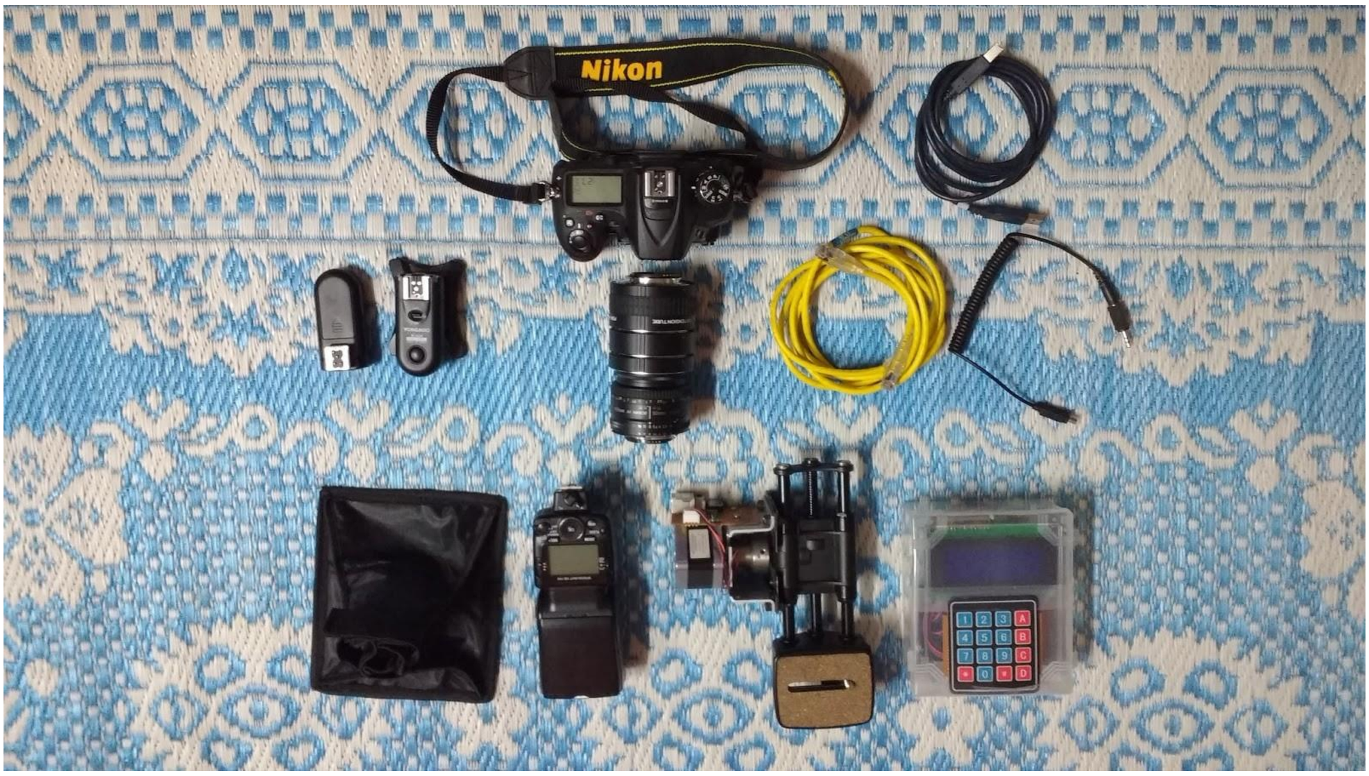
1. The Stepper motor had to be controlled with precision.
2. The keypad for handling user input.
3. The display unit for giving feedback to the user.
4. Control the camera for taking a photo.

Since I have had no experience with this, I had to start from scratch ! So I decided to take multiple sprints, so to say, to establish each interface. Once I was comfortable with each one, I started the integration of multiple interfaces. First Display - Keypad, then Display - Keypad - Stepper Motor and finally Display - Keypad - Stepper Motor - Camera Triggering.

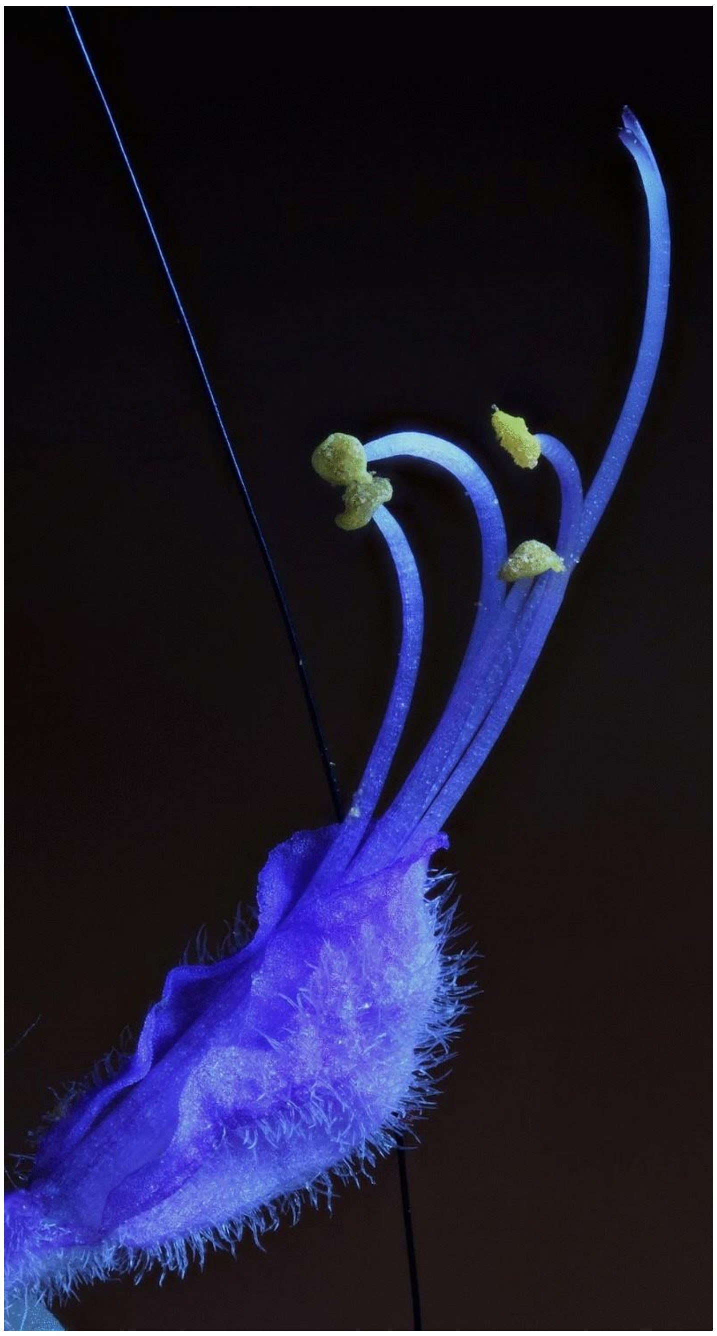
Challenges:

1. Mounting the stepper motor to the rail without misalignment and needed multiple rebuilds and concepts. misalignment will lead to missed steps and inaccuracies.
2. Soldering ! During the initial proto phase, everything was on a breadboard which was good for playing around with multiple possibilities. But then you will occasionally get some loose contact because of the breadboard. Hence had to move to proto-boards. Since I didn't had any prior experience of soldering, I ended of doing one board multiple times till it was stable.
3. Since there were multiple interfaces, I had to optimise the handling of each one for better response. Things like, deciding the screen refresh rate to save processing and avoiding flickering, how to manage keyboard interruptions during the sequence were some of the main ones.

The typical setup involves:



Below image is another result of this automation. Please note that the black thin object is a human hair!



Conclusion:

Irrespective of the frustrations in between, the sleepless nights of debugging and learning and other challenges, the joy of seeing it to work is unmatched.

If I, without any prior knowledge, can do this project, it means only one thing. Anyone can do it, thanks to the simplicity of the Arduino interface, the hardware options available for each use cases and the learning material available online. Only need is an idea and some perseverance.