NICIS
NATIONAL INTEGRATED CYBERINFRASTRUCTURE SYSTEM
CHPC

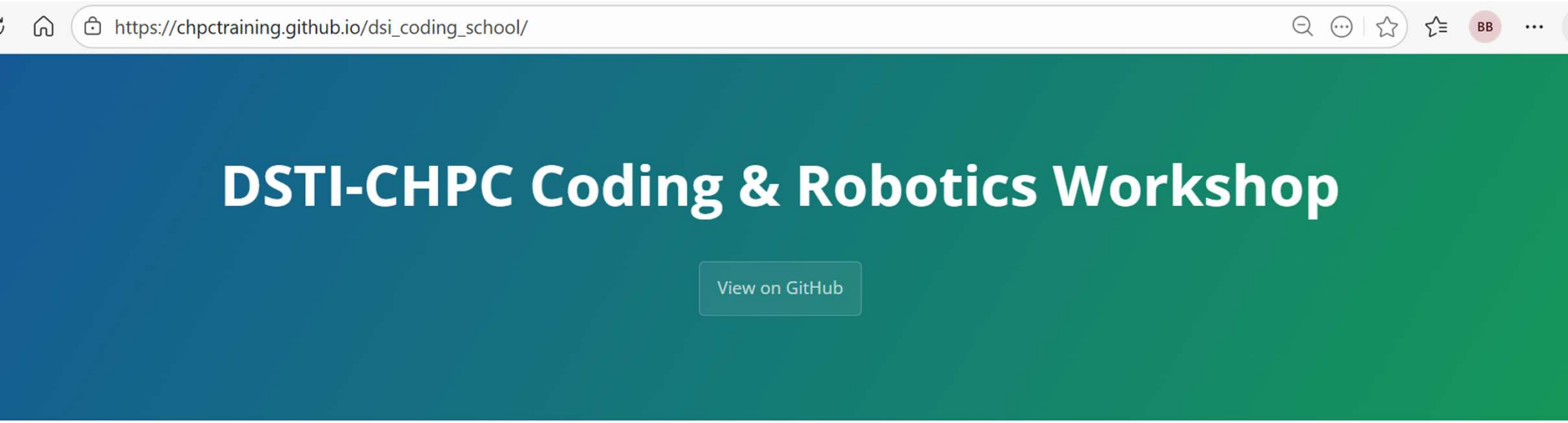
DSTI-CHPC Coding & Robotics Workshop

Introduction to Robotics 1

A national initiative of the Department of Science and Innovation and implemented by the CSIR

science & innovation
Department:
Science and Innovation
REPUBLIC OF SOUTH AFRICA

CSIR
Touching lives through innovation



DSTI-CHPC Coding & Robotics Workshop

The Centre for High Performance Computing (CHPC), funded by theDepartment of Science, Technology and Innovation (DSTI) of the South African government, has launched a three-year flagship project aimed at raising awareness of coding and robotics within the Department of Basic Education and the general public. The project began in 2022 and ran until the end of March 2025. A new three-year project started in 2026. It involves training coding and robotics subject advisors and officials from all districts across South Africa's nine provinces. The five-day training program covers key areas such as:

- Coding using Scratch software.
- An introduction to robotics and programming robots with Scratch.
- Arduino and Microbit Introduction
- Electronics Introduction

Time	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5
08:45 - 9:00	Registration	Registration	Registration	Registration	Registration
9:00 -10:30	Opening: Computer Hardware and Coding for DSTI Workshop	Scratch Coding Session 3	Robotics Practical Session 1	Microbit Session 1	Microbit Session 3
10:30 - 11:00			**Tea Break**		
11:00 - 13:00	Scratch Coding Session 1	Scratch Coding Session 4	Robotics Practical 2	Microbit Session 2	Microbit Session 4
13:00- 14:00			**Lunch**		
14:00 -15:00	CHPC Facility Tour	Scratch Coding Session 5	Robotics Practical Session 3	{codeclub} Session 1	Feedback and Closing
15:30 - 16:00			**Tea Break**		
16:00 - 17:00	Scratch Coding Session 2	Scratch Coding 6	Robotics Practical Session 4	{codeclub} Session 2	Departure

NICIS NATIONAL INTEGRATED CYBERINFRASTRUCTURE SYSTEM CHPC

science & innovation Department: Science and Innovation REPUBLIC OF SOUTH AFRICA

CSIR Touching lives through innovation

Let’s Get Started With Robotics!

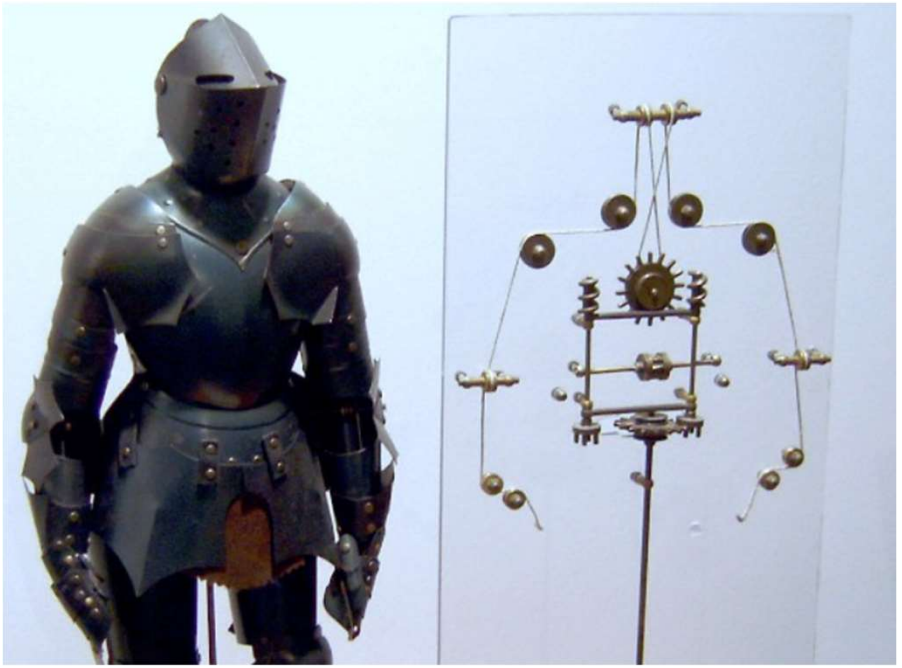
What is a Coding?

What is a Robot?

What is a Robot?

a **robot** can be defined as a machine designed to **automate tasks traditionally performed by humans**.

What is a Robot?

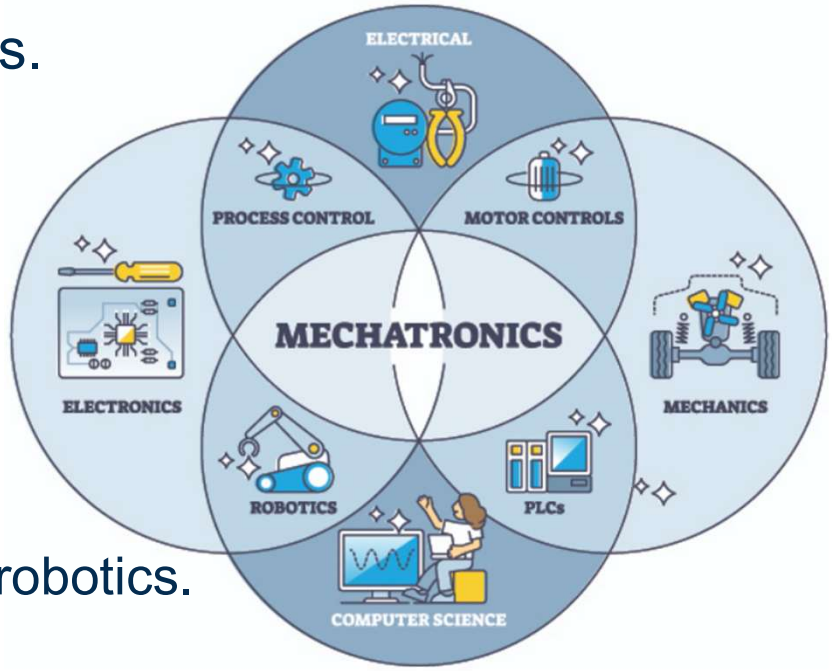


Around 1495, Leonardo di Vinci drew up plans for a mechanical knight – an armour-clad ‘robot’ that could sit up, move its head, and even wave a sword in its hands.

Theory!

What is Robotics?

Definition: Robotics is the interdisciplinary field that involves the design, construction, and operation of robots.



Core Components:

- **Mechanics**: Structure and movement
- **Electronics**: Power and control systems
- **Programming**: Logic and decision-making
  - Like following a cooing recipe

The field of **mechatronics** is closed related to robotics.

Mechanics In Robotics

Definition: Mechanics refers to the physical structure of the robot—its body, limbs, joints, and gears.

Key Components:

- Frames and Chassis: The backbone of the robot.
- Motors and Actuators: Create movement.
- Wheels/Tracks/Legs: Provide mobility.
- Arms/Grippers: Allow the robot to interact with objects.

Example: A robotic arm uses motors to control the joints, allowing it to pick up objects.

Electronics (Electrical) In Robotics

Definition: Electronics provide power and control to the mechanical parts.

Key Components:

- Power Supply: Battery or external power source.
- Microcontroller/Microprocessor: Acts as the brain of the robot
- Motors and Drivers: Convert electrical energy into motion.
- Wiring & Circuits: Connect all components.

Example: A motor driver controls the speed and direction of a DC motor based on commands from the microcontroller.

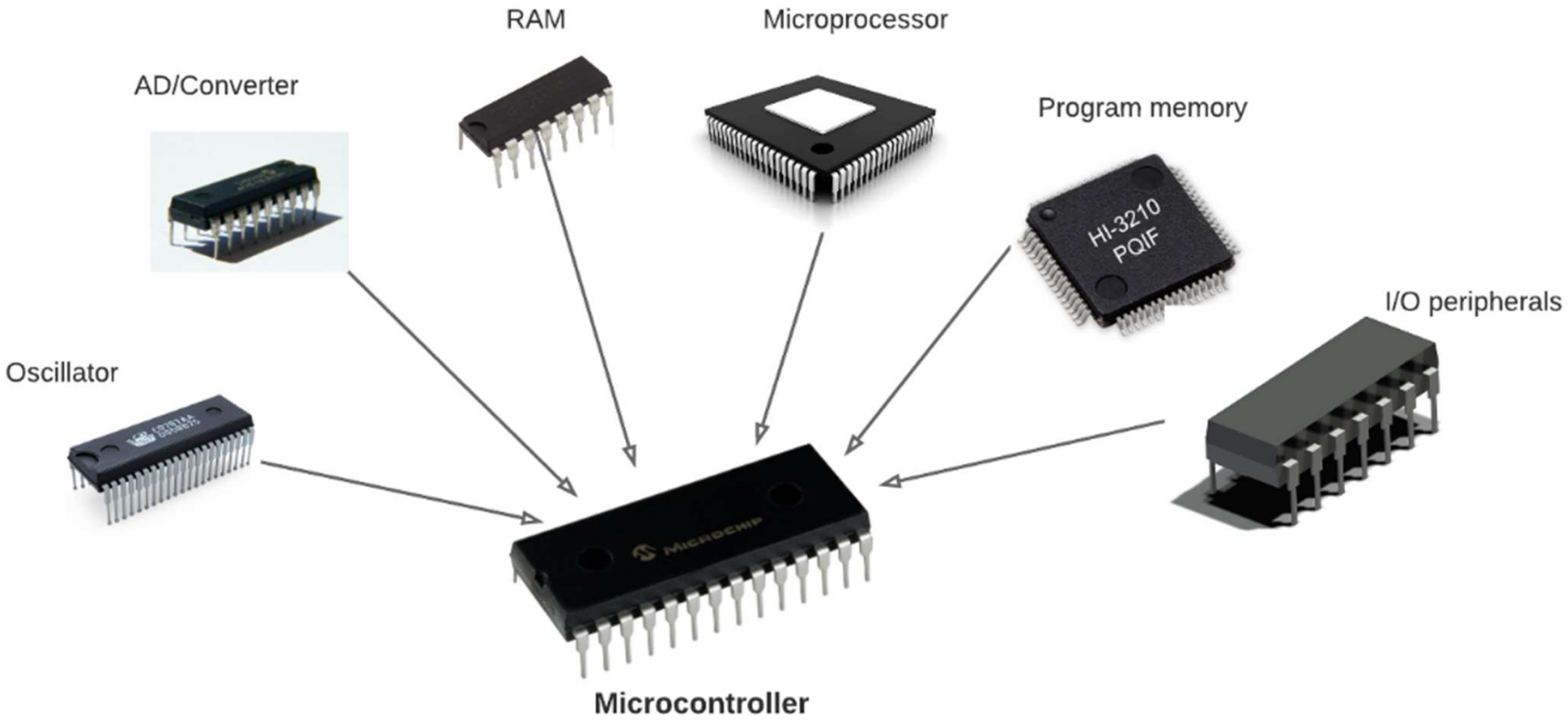
Microcontroller

Definition: A microcontroller is like small computer on a single integrated circuit (IC) designed to control devices or processes.

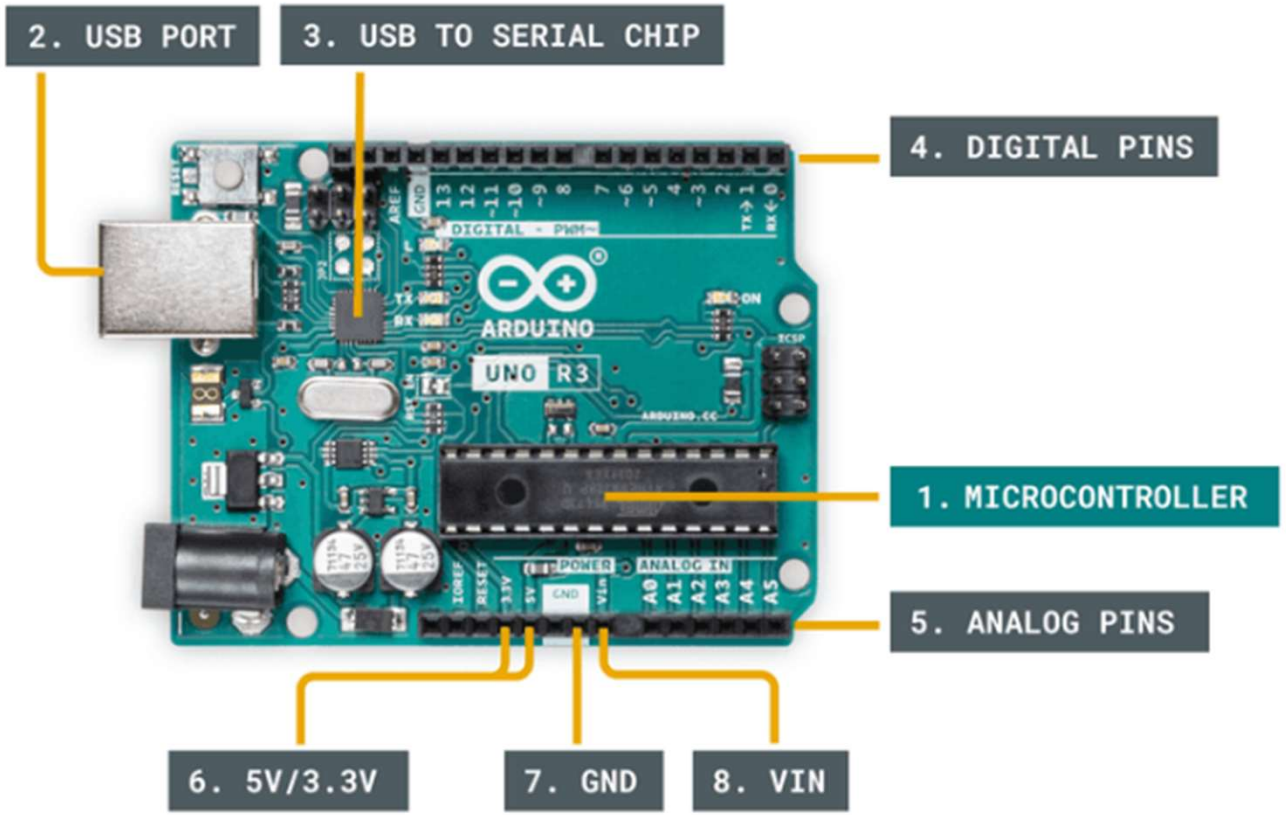
Basics:

- **CPU**: The brain of the microcontroller that executes programs and instructions.
- **Memory:** Used to store instructions and information
- **I/O Pins:** Connect to external devices, such as sensors, motors, or displays, allowing the microcontroller to interact with the physical world.

Microcontroller



Electronics: The Microcontroller (Arduino)



Key components of an Arduino board.

Electronics: The Microcontroller (Arduino)

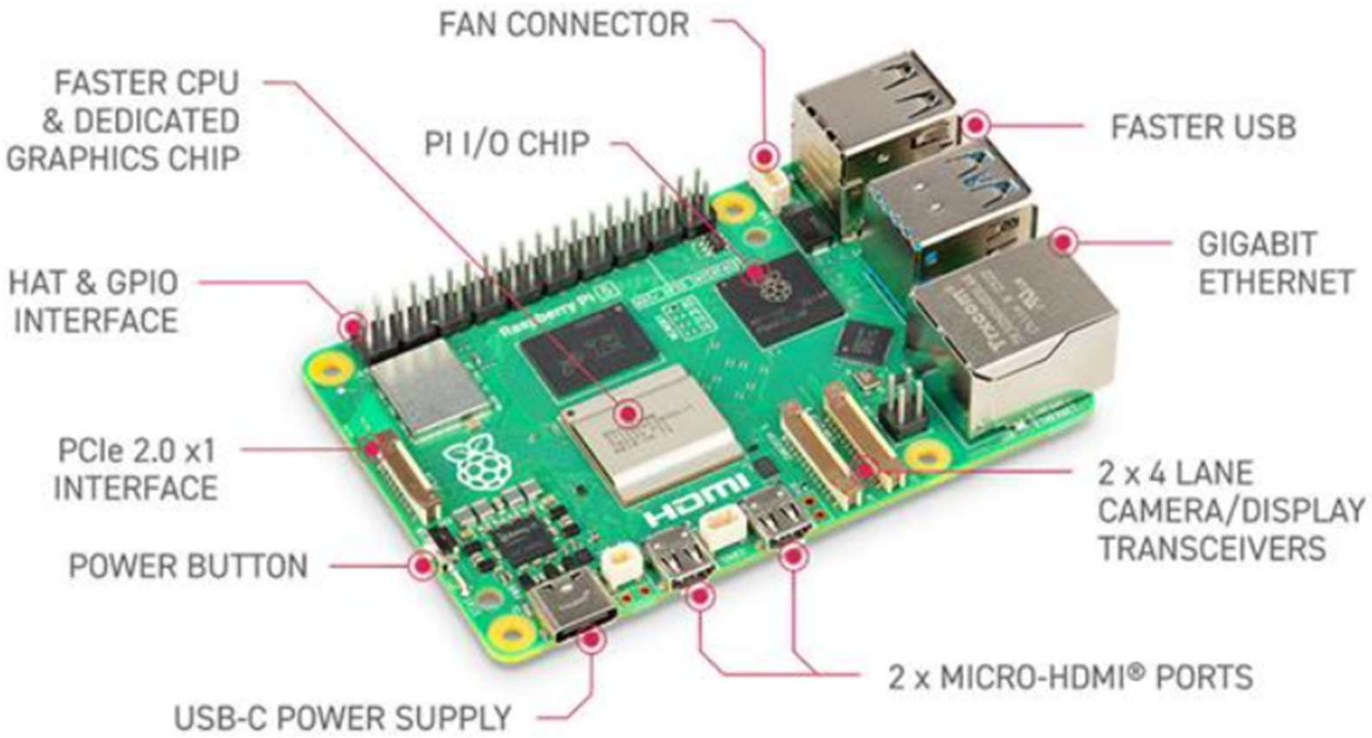
- **1. Microcontroller** - this is the brain of an Arduino, and is the component that we load programs into. Think of it as a tiny computer, designed to execute only a specific number of things.
- **2. USB port** - used to connect your Arduino board to a computer.
- **3. USB to Serial chip** - the USB to Serial is an important component, as it helps translating data that comes from e.g. a computer to the on-board microcontroller. This is what makes it possible to program the Arduino board from your computer.
- **4. Digital pins** - pins that use digital logic (0,1 or LOW/HIGH). Commonly used for switches and to turn on/off an LED.
- **5. Analog pins** - pins that can read analog values in a 10 bit resolution (0-1023).
- **6. 5V / 3.3V pins**- these pins are used to power external components.
- **7. GND** - also known as `ground`, `negative` or simply `-`, is used to complete a circuit, where the electrical level is at 0 volt.
- **8. VIN** - stands for Voltage In, where you can connect external power supplies.

Microcontroller vs Microprocessor

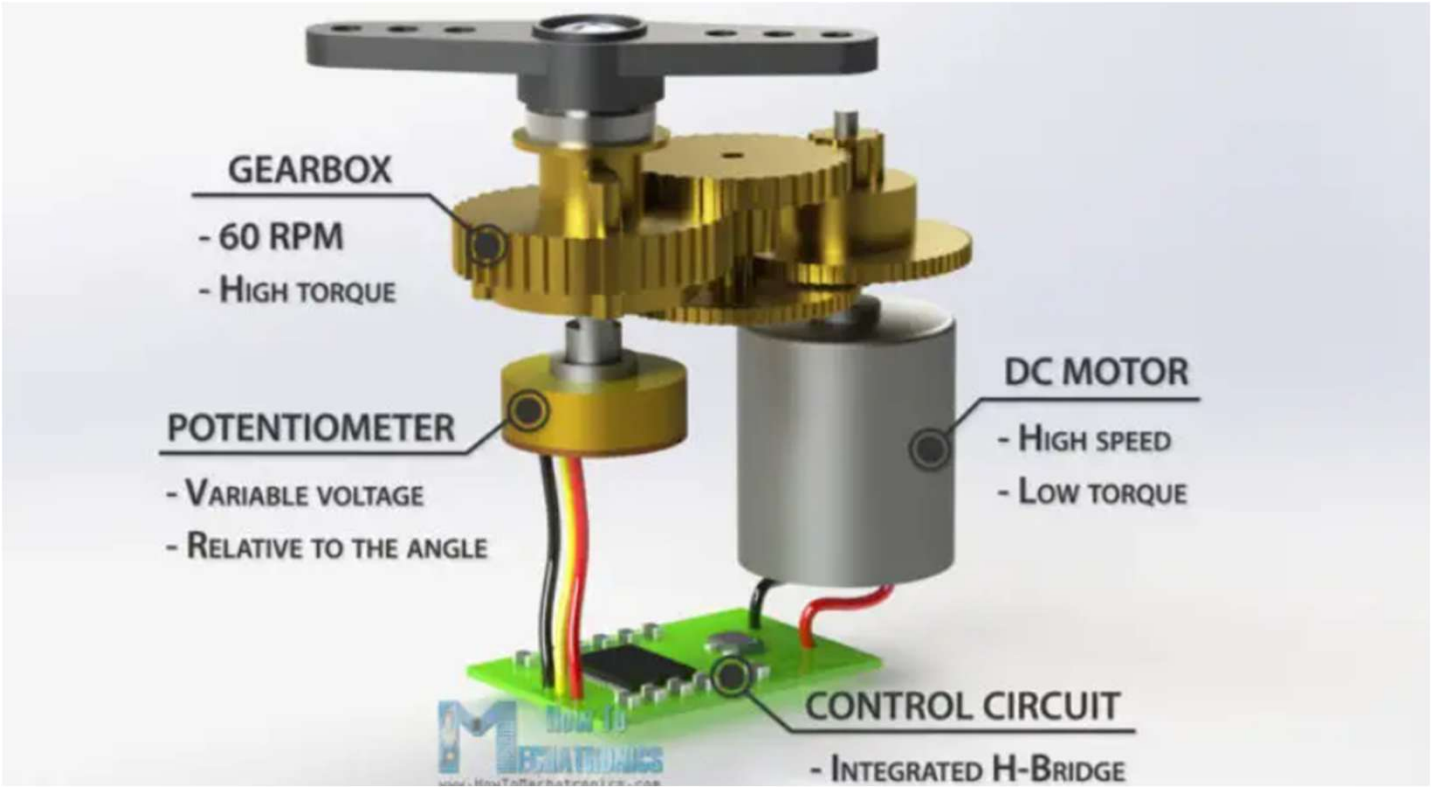
Microcontrollers are self-contained systems with all components needed for control in a single chip. They are optimized for controlling tasks and are found in everyday devices like microwaves, thermostats, and toys.

Microprocessors, in contrast, are more powerful but require external components (like RAM and storage) to function. These are used in cell phones, computers like desktops and laptops.

Microprocessor/CPU on Raspberry Pi Board



Servo Motor


GEARBOX
- 60 RPM
- HIGH TORQUE
POTENTIOMETER
- VARIABLE VOLTAGE
- RELATIVE TO THE ANGLE
DC MOTOR
- HIGH SPEED
- LOW TORQUE
CONTROL CIRCUIT
- INTEGRATED H-BRIDGE

Servo Motor

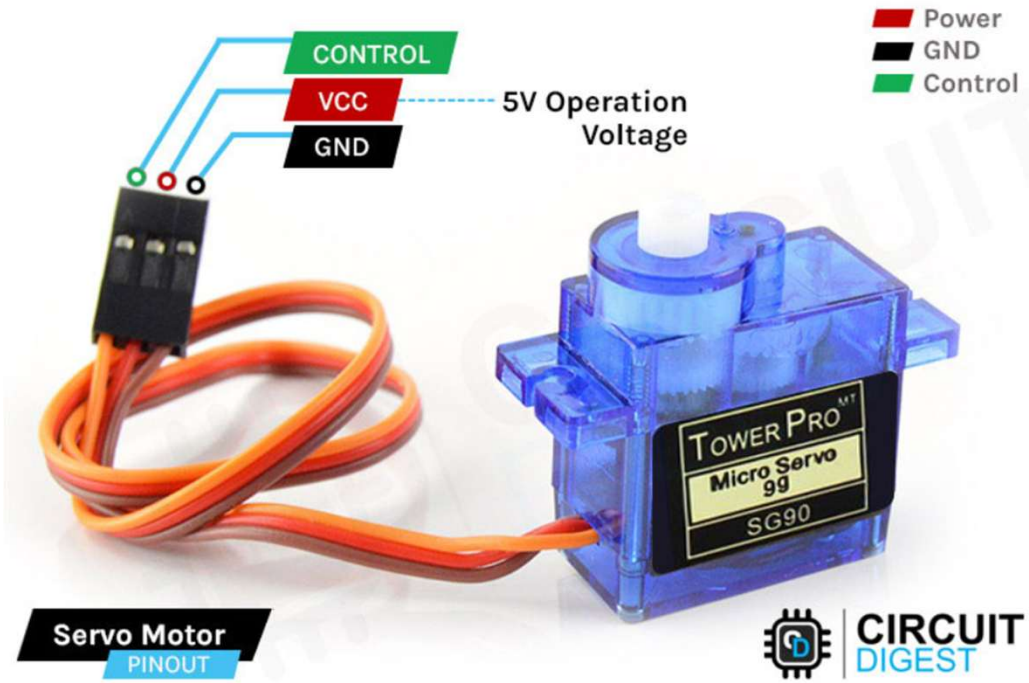
- A servo motor is a type of **motor** that provides **precise** control of **angular** or **linear** position, speed, and torque.
- Unlike regular motors, which simply **rotate** continuously, a servo motor can be instructed to move to a **specific** position and hold that position until further instructions are given.
- This makes it highly useful in applications requiring **accurate** movement and control, such as in **robotics**, **drones**, and automation systems.

Servo Motor

SG90 Micro Servo technical specifications:

Stall Torque	1.2kg·cm @4.8V, 1.6kg·cm @6V,
Operating Voltage	3.5 – 6V
No Load Current	100mA
Stall Current	650mA
Max Speed	60 degrees in 0.12s
Weight	9g

Servo Motor



- **GND** Ground (Brown Wire) – This is the ground pin.
- **VCC** +5V (Red Wire) – Voltage is supplied to the servo motor through this Pin.
- **Control** (Orange Wire) – Through this wire, Position control signals are received via PWM.

Practical!

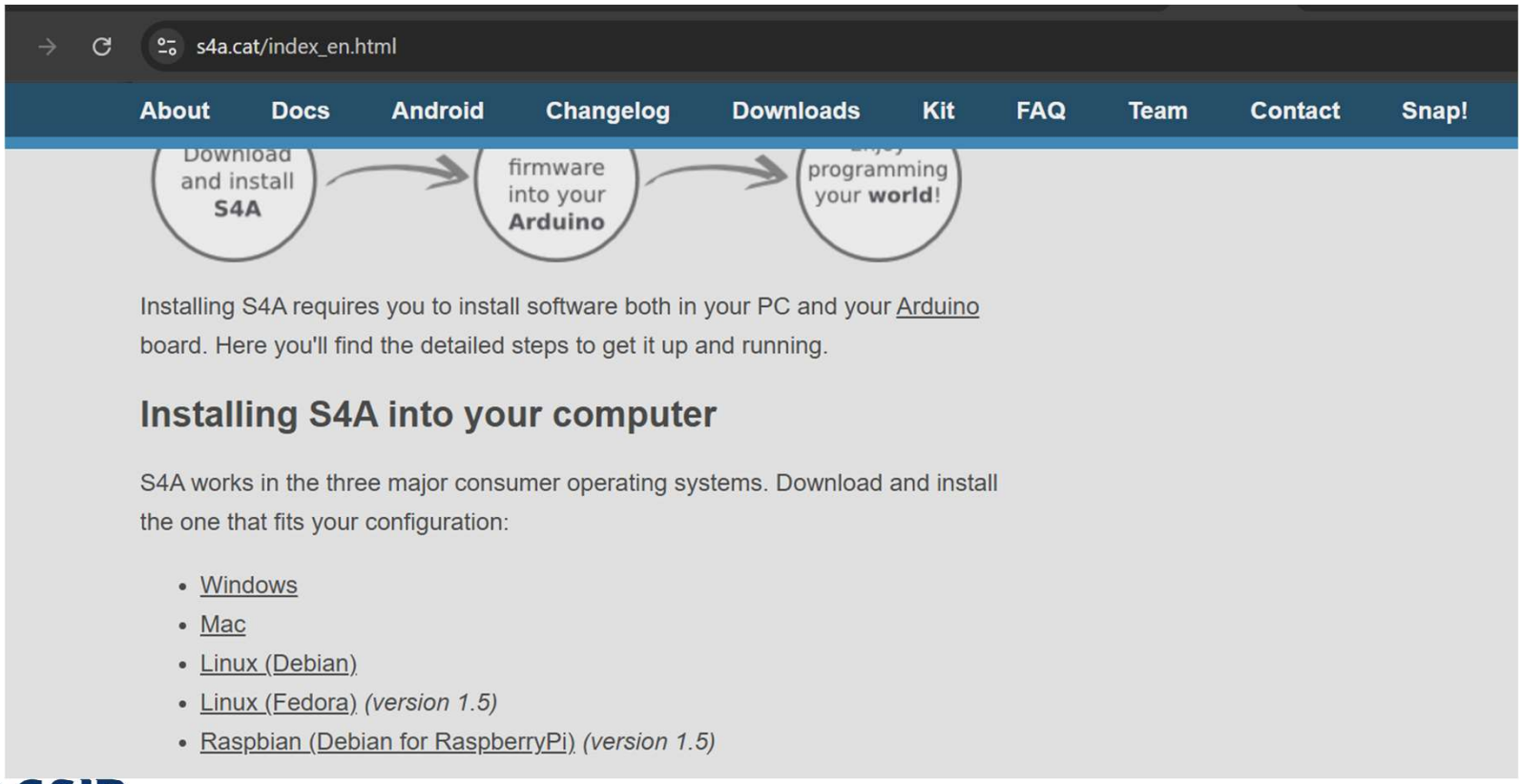
Using CHPC Laptops!

Installing Software

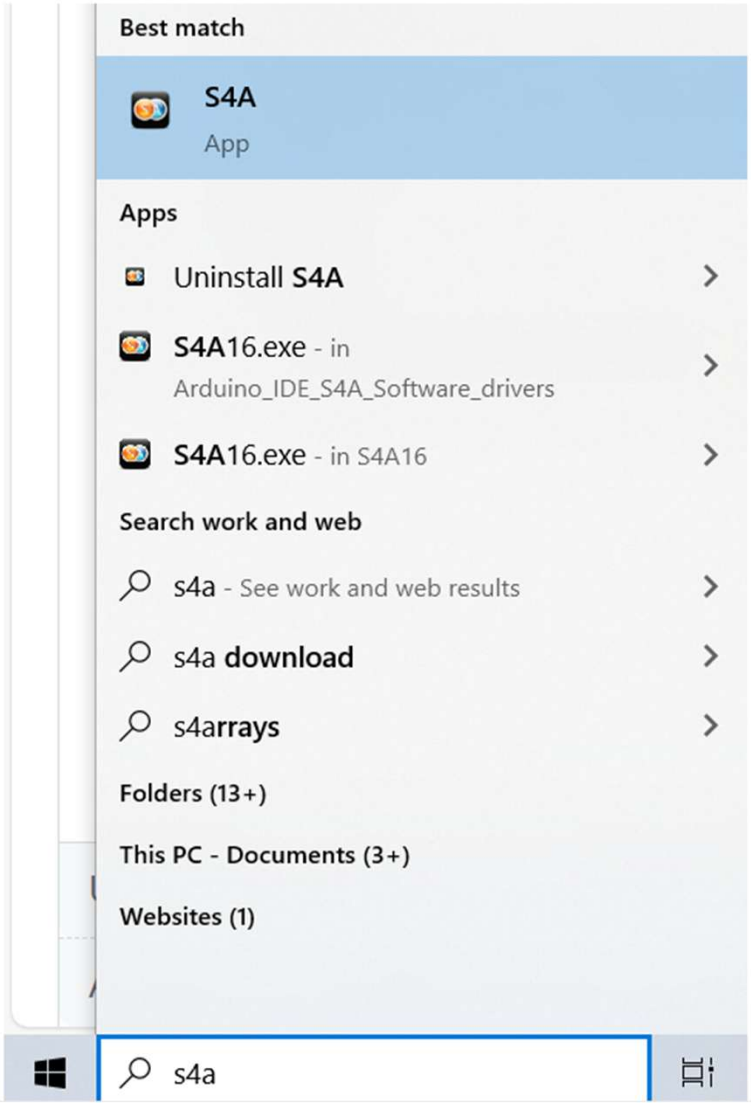
Step 1 Install Arduino IDE: https://www.arduino.cc/en/software/

Installing Software

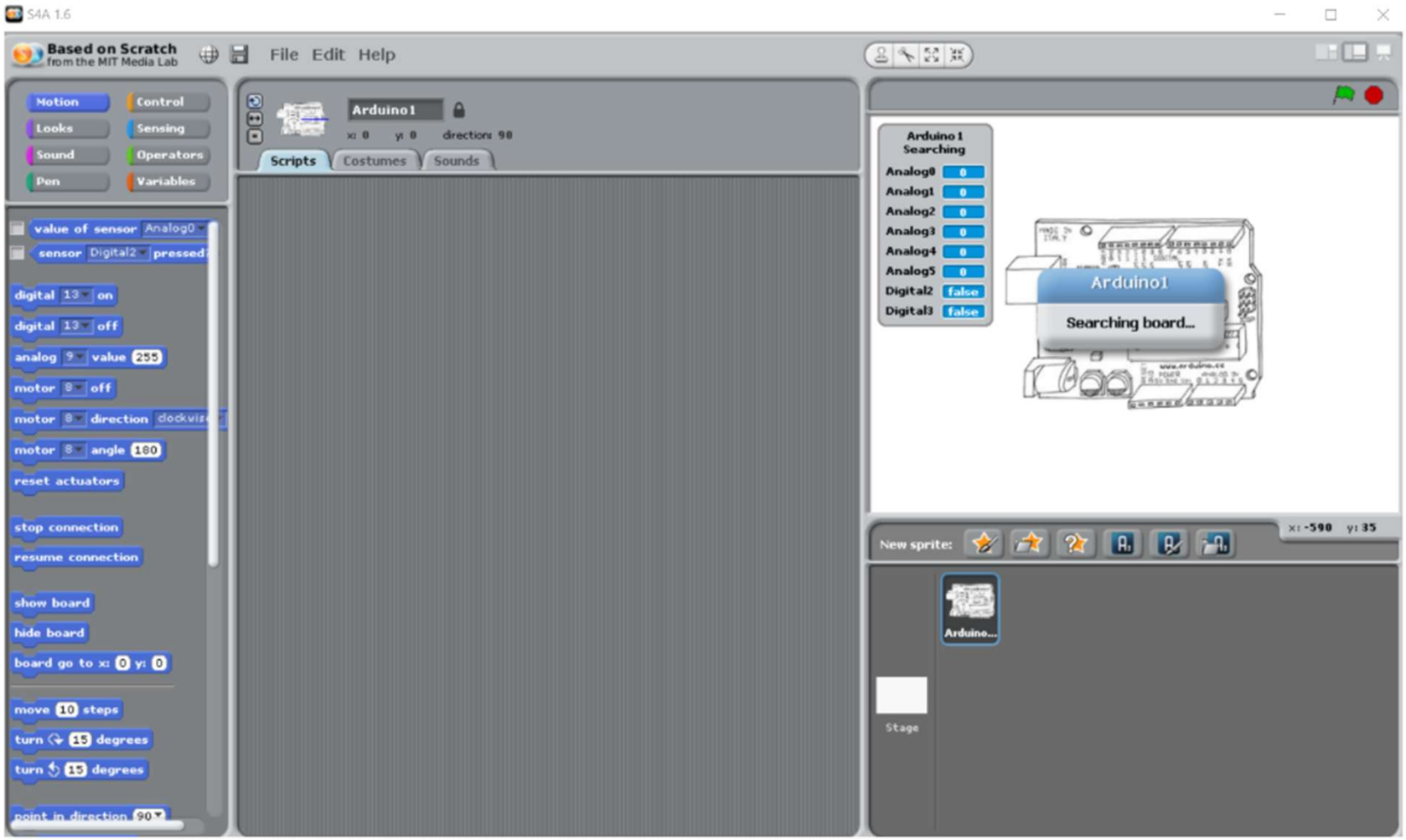
Step 2 Install S4A: https://s4a.cat/downloads/S4A16.zip



1. Open up S4A (Scratch for Arduino) in Windows

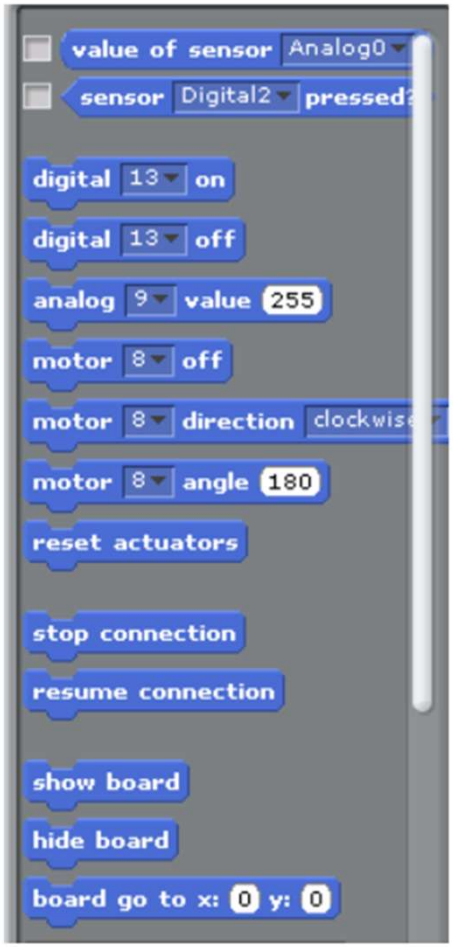


2. You should see the following:

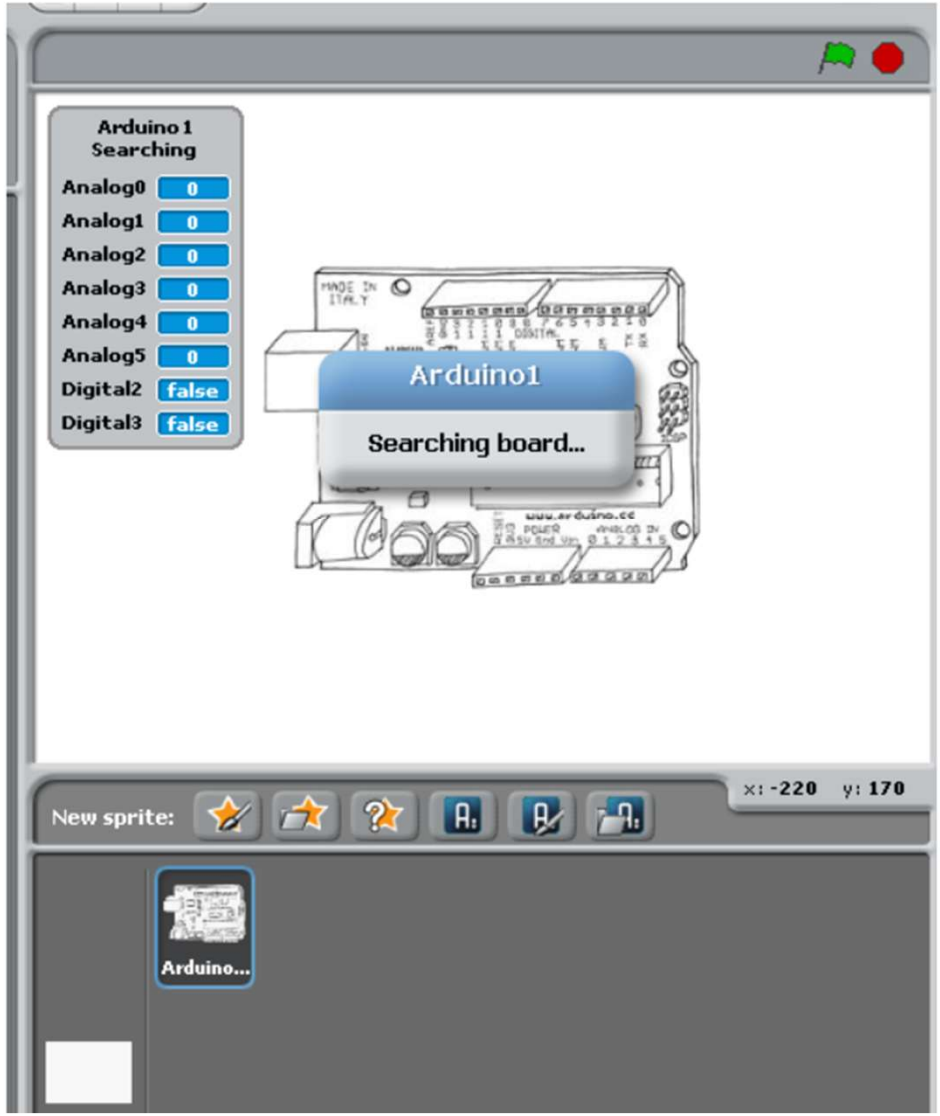


We will only be working with Motion and Control:

You will notice Motion has different blocks:

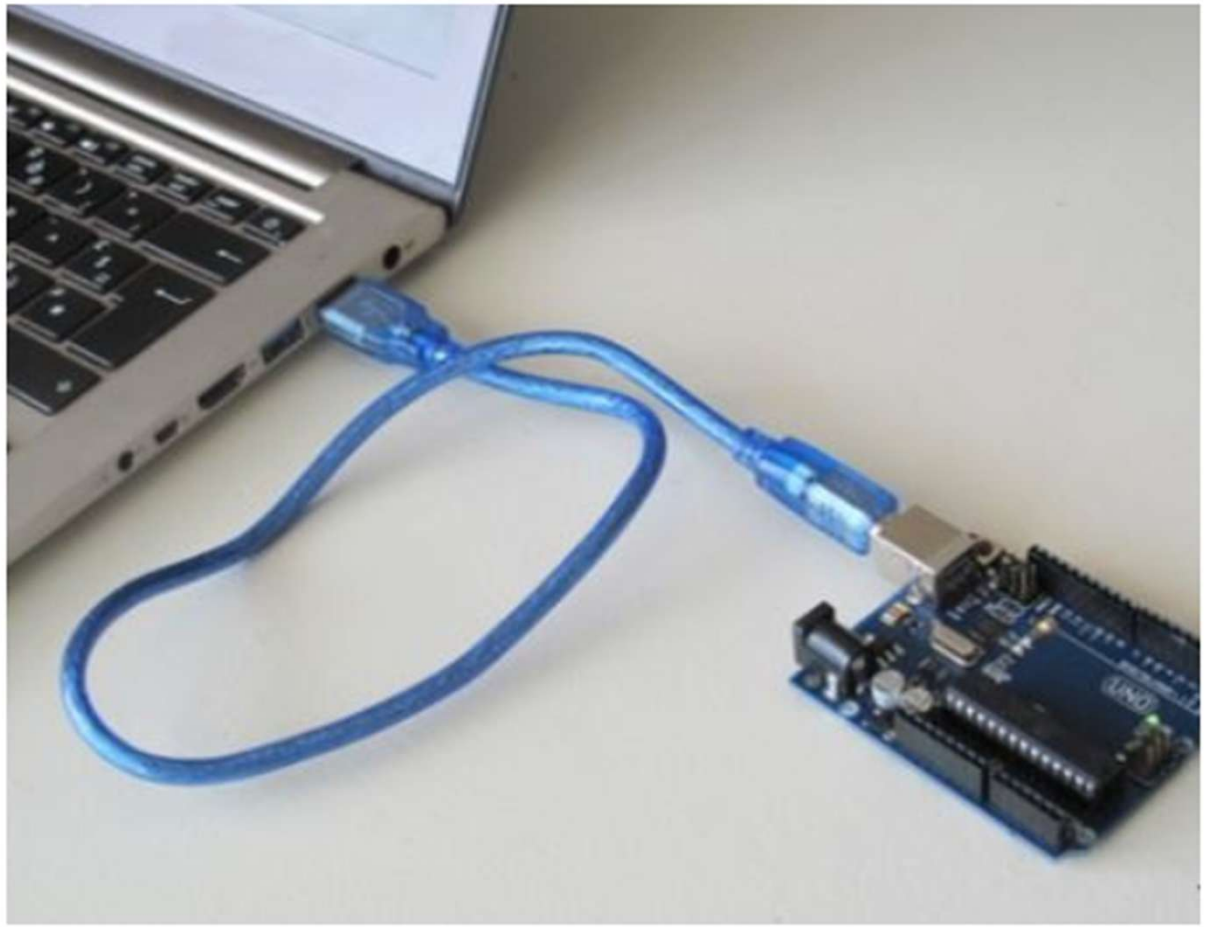


You will also see this window on the right, which shows the Arduino board:

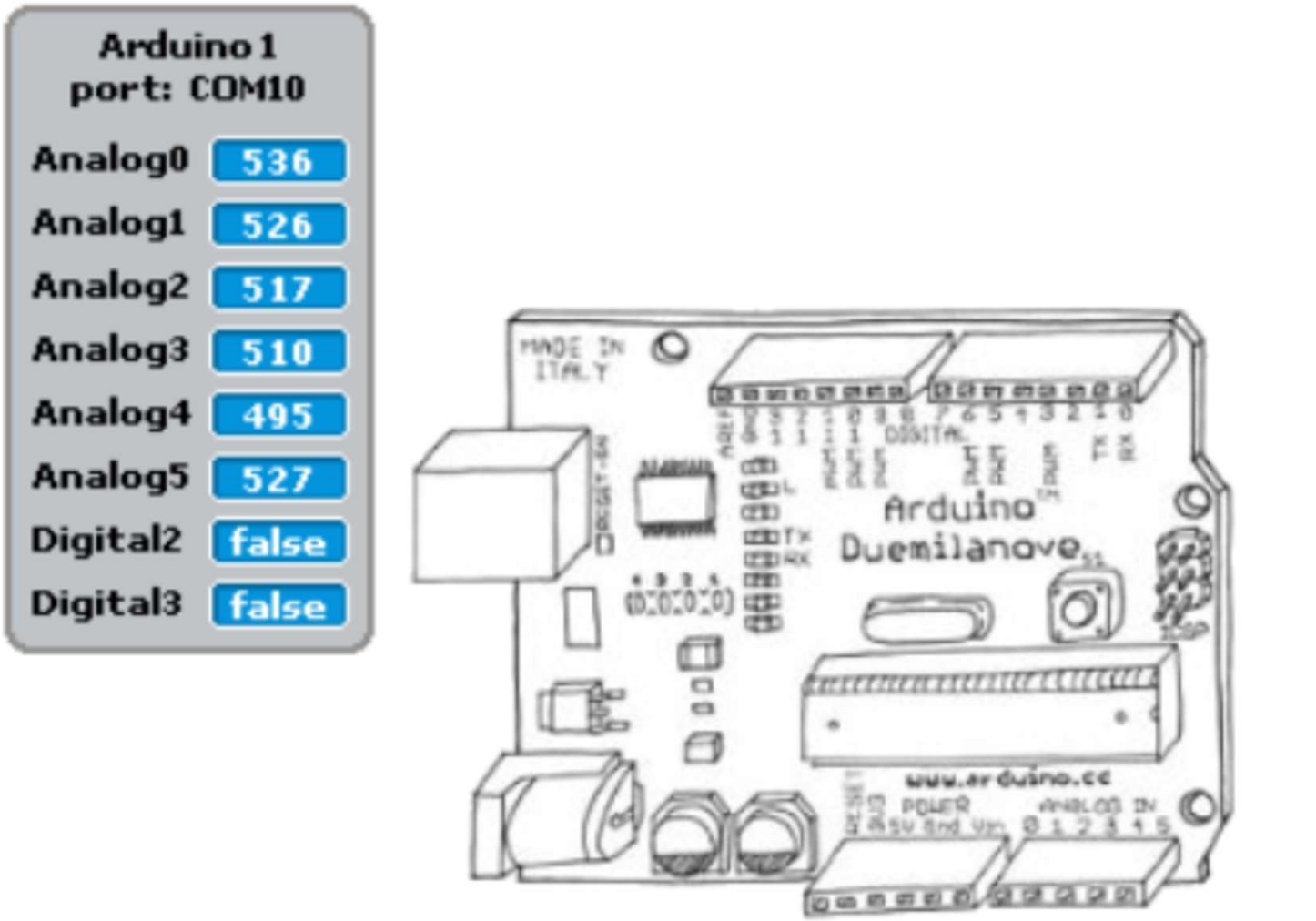


However, it says "Arduino1, Searching board...", its trying to connect to your Arduino:

So let us connect it:



You should see some flashing lights on your Arduino board, and now in Scratch you should see:



What is important to see is all those text fields have numbers changing all the time and it says "port: COM10" it may be a different on you laptop like "port: COM5" or "port: COM3", this just tells us that Scratch is now connected to your Arduino!

Now you should all be quite familiar with Scratch, so now create the following:

Then run it:

You should have seen that on your board, one of the LED lights is flashing on and off.

Modify the "wait" time

Motor Control!

Make sure the USB

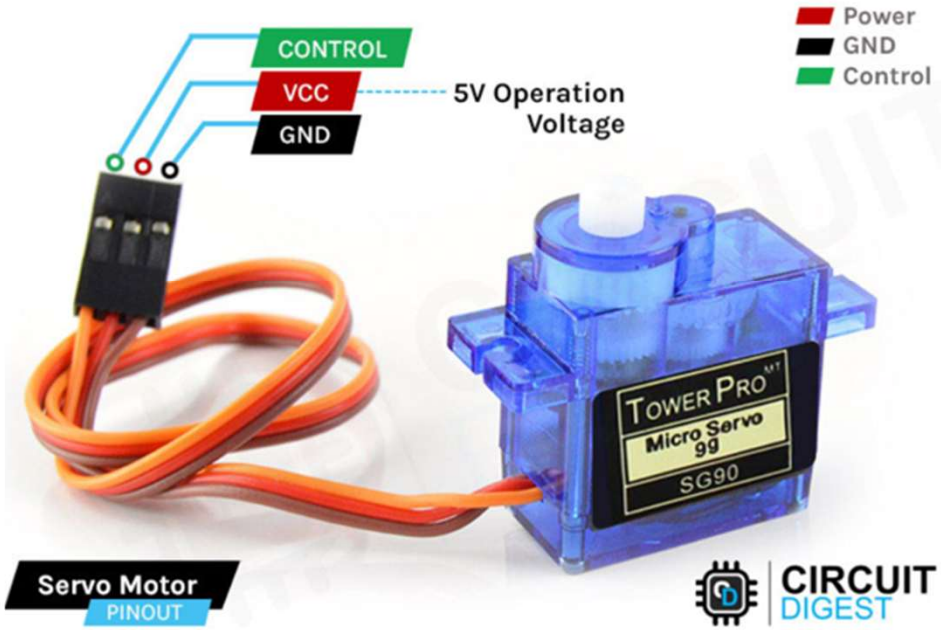
Cable Is Not Connected

Now let us connect the two servo motors to the Arduino.

NOTE: Please do not try to turn the motor with your hand, you can break it!

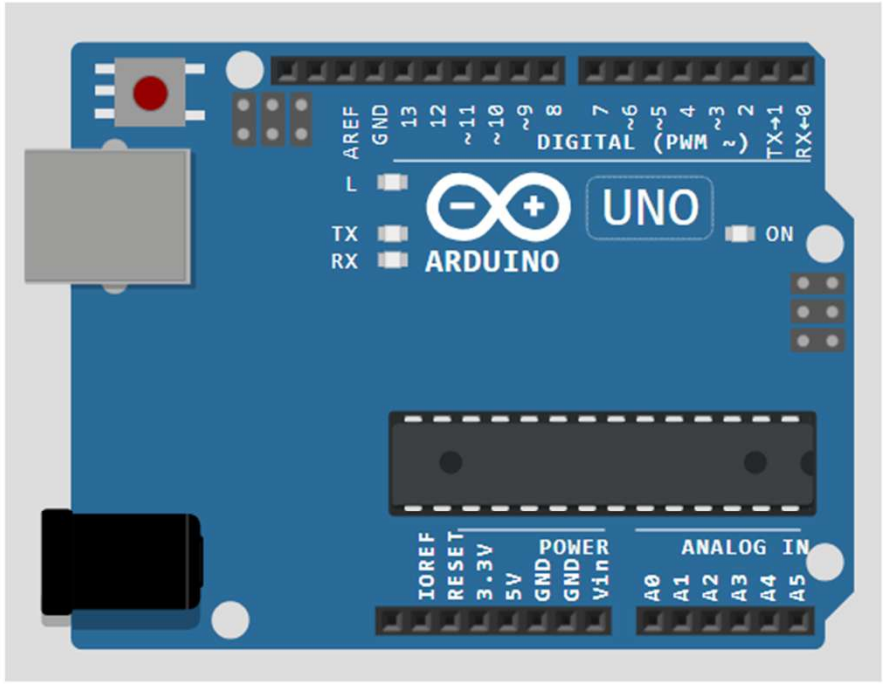
We have our Arm motor and Gripper motor.

Remember the wiring:

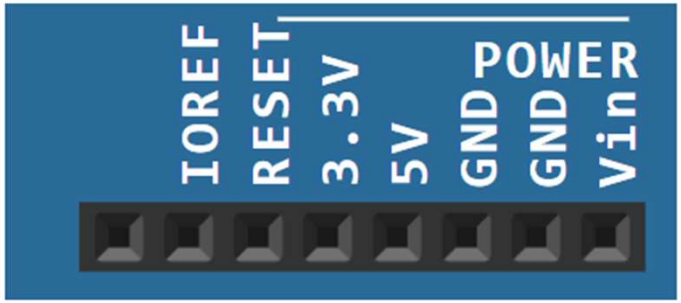


- **GND** Ground (Brown Wire) – This is the ground pin.
- **VCC** +5V (Red Wire) – Voltage is supplied to the servo motor through this Pin.
- **Control** (Orange Wire) – Through this wire, Position control signals are received via PWM.

Now your Arduino looks something like this:



We need to power both servos with these pins:



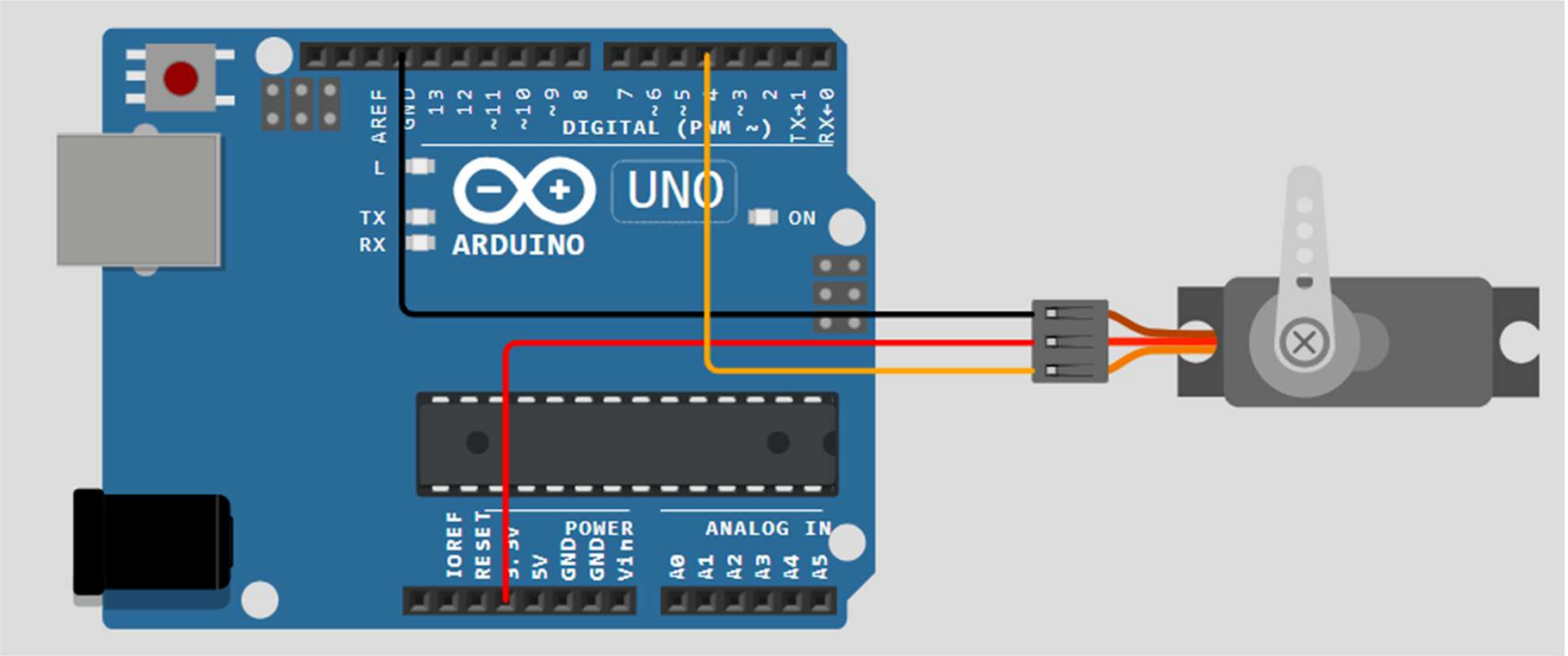
And we will control the angle with these pins:

When we learnt about servo motors we saw they require a DC voltage of:

Operating Voltage	3.5 – 6V

Using the male to male jumper wires:

- Connect the 5v pin from your Arduino to the red connector on your Gripper motor
- Connect a GND (ground) pin from your Arduino to the brown connector on your Gripper motor
- Connect pin 4 from your Arduino to the orange connector on your Gripper motor

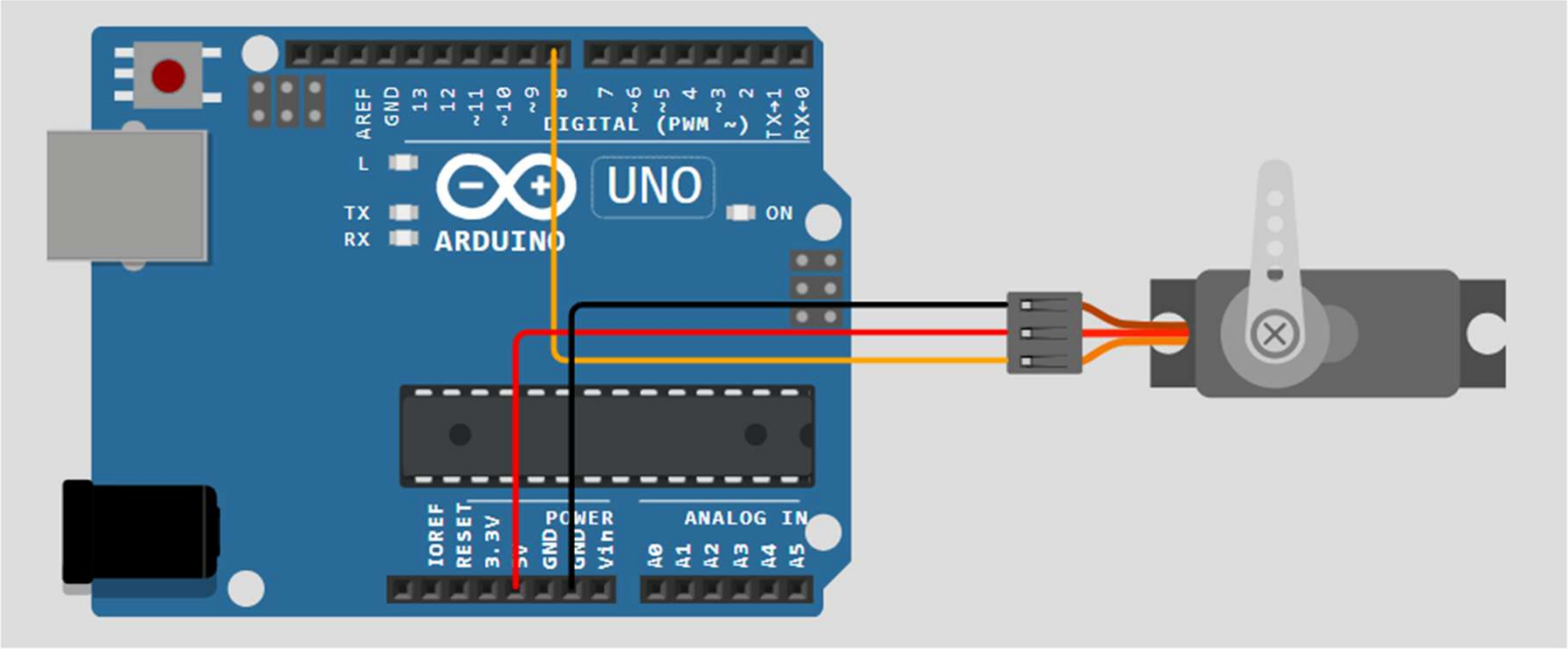


Next let us do the Arm motor.

Using the male to male wires:

- Connect the 5V pin from your Arduino to the red connector on your Arm motor
- Connect a GND (ground) pin from your Arduino to the brown connector on your Arm motor
- Connect pin 8 from your Arduino to the orange connector on your Arm motor

So you should end up with the following:



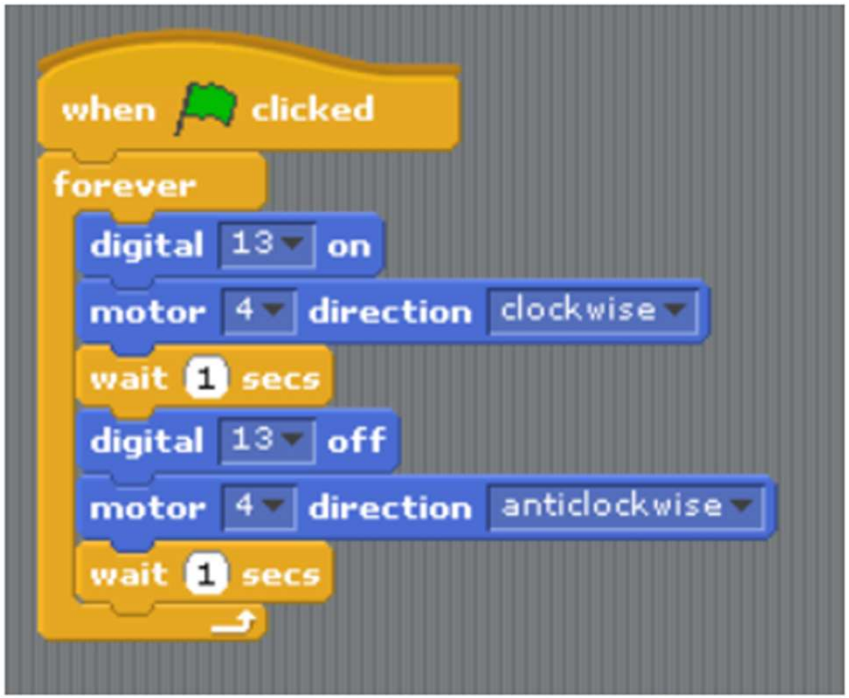
Once the motor wires are connected – reconnect the USB Cable

Scratch Code

Now that you have connected your motors let us test them out.

Motor Control

Create the following Scratch code blocks:



What does it do?

Change the motor value from "4" to "8" to control the other motor.

Make sure you have tested both of your motors before going on to the next section.

Angle Control

Next, create a new project but now use this motor block:

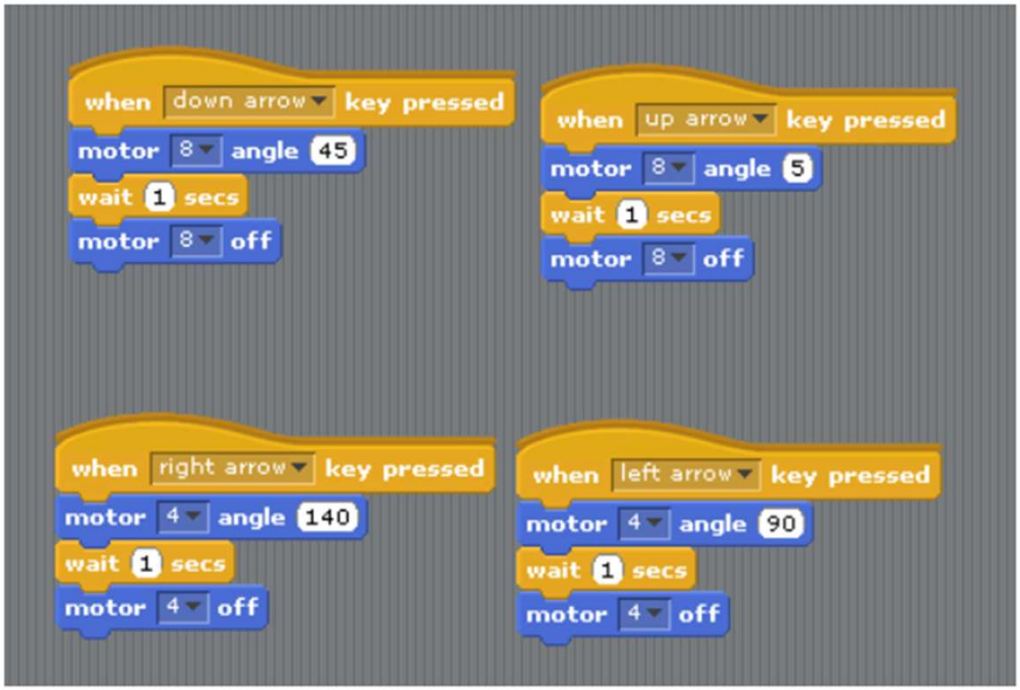
Change the values to see the angle of the servo change.

Make sure you have tested both of your motors.

Control with Arrow Keys

Now try to control both motors with arrow keys.

Build the following Scratch code:

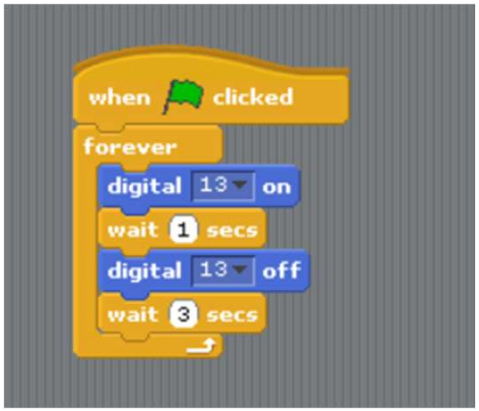


The "wait" control block is important stabilize the motor.

Theory!

Programming In Robotics

Definition: Programming is the process of writing a set of instructions, known as code, that a computer or other devices like microcontrollers follow to perform specific tasks. These instructions are written in programming languages such as Python, C++, or even visual languages like Scratch.



Key Components:

- Programming/Coding Languages: C/C++, Python, Scratch
- Control Structure: Loops and conditionals
- Functions: Reusable code
- Sensor control: Real time processing of data from
- environment.

Example: A robot programmed to follow a line uses a loop to constantly check sensor data and adjust movement.

Sensors In Robotics

Definition: Sensors provide robots with information about their environment.

Types of Sensors:

- Distance Sensors: Measure proximity
- (e.g., ultrasonic, infrared).
- Light Sensors: Detect light and color.
- Touch Sensors: Respond to physical pressure.
- Gyroscope/Accelerometer: Measure orientation and movement.

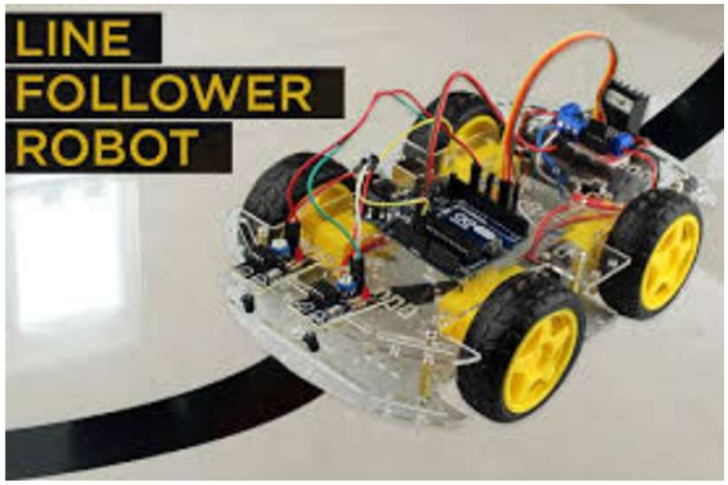
Example: An ultrasonic sensor helps a robot avoid obstacles by detecting objects in its path.

Combining Mechanics, Electronics, and Coding

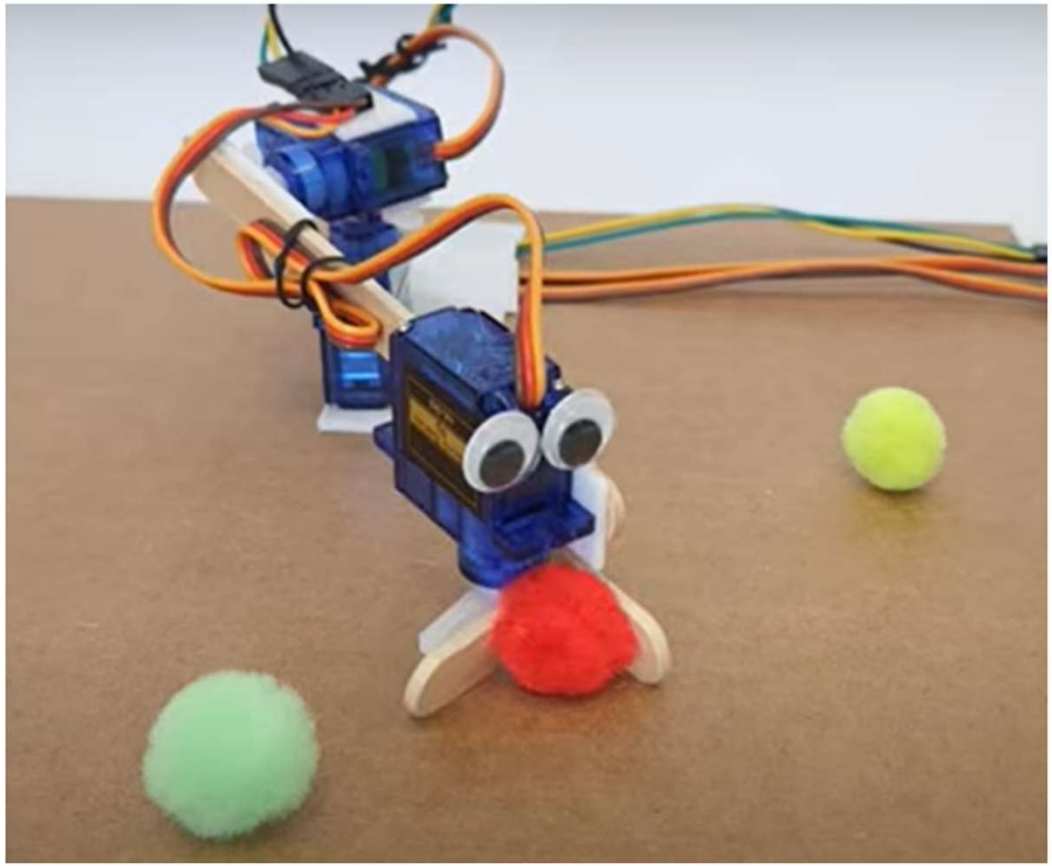
Example: A line-following robot:

- Mechanics: The robot's chassis and wheels provide mobility.
- Electronics: Motors and sensors are wired to a microcontroller.
- Programming: Code continuously reads sensor data and adjusts the motor speed to stay on track.

Real-world Applications: Autonomous cars, industrial robots, drones.



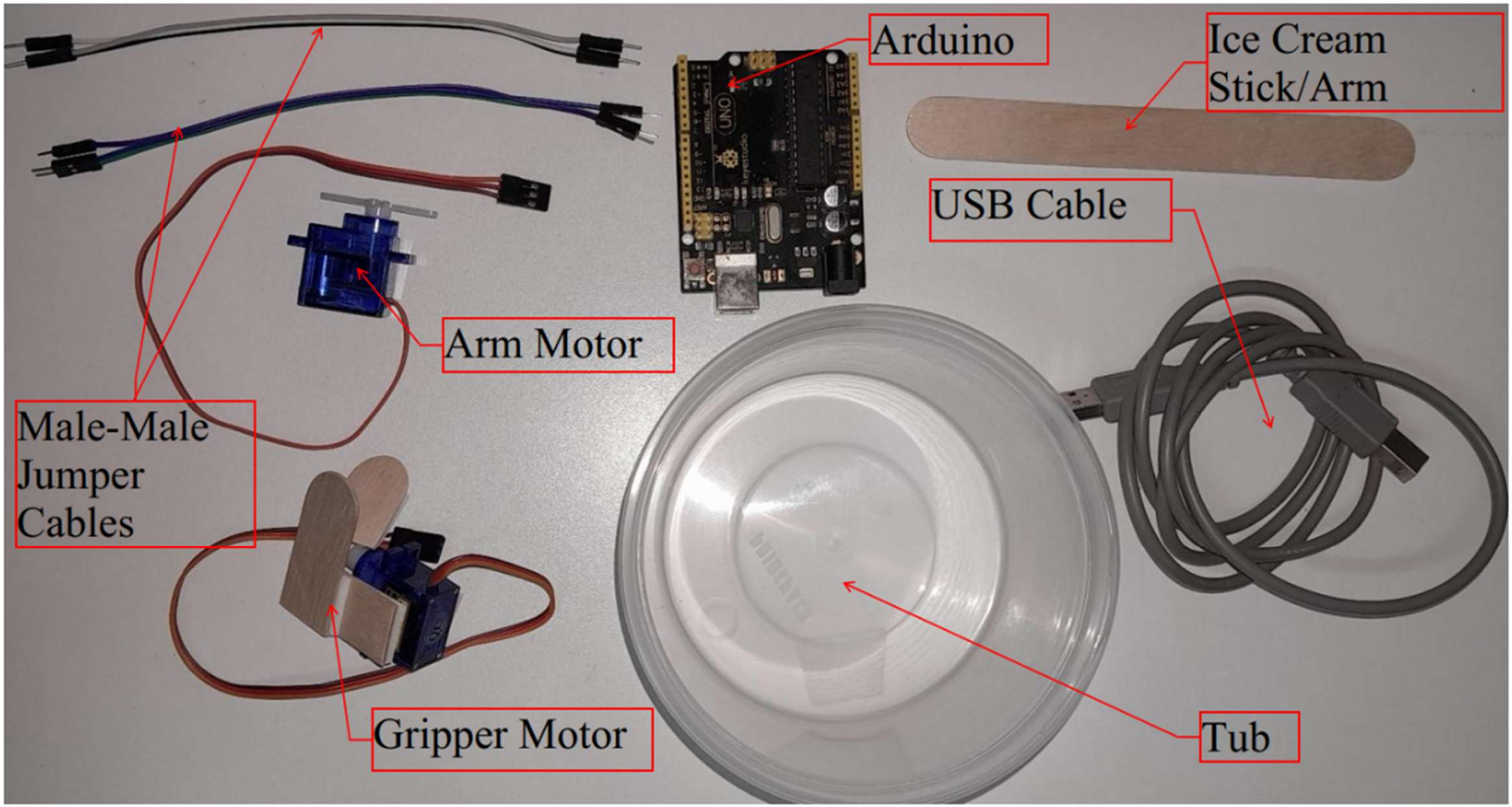
Building A Robotic Arm!



Objective: Build a robotic arm that can grab a small fluff ball and lift it using your laptop keys.

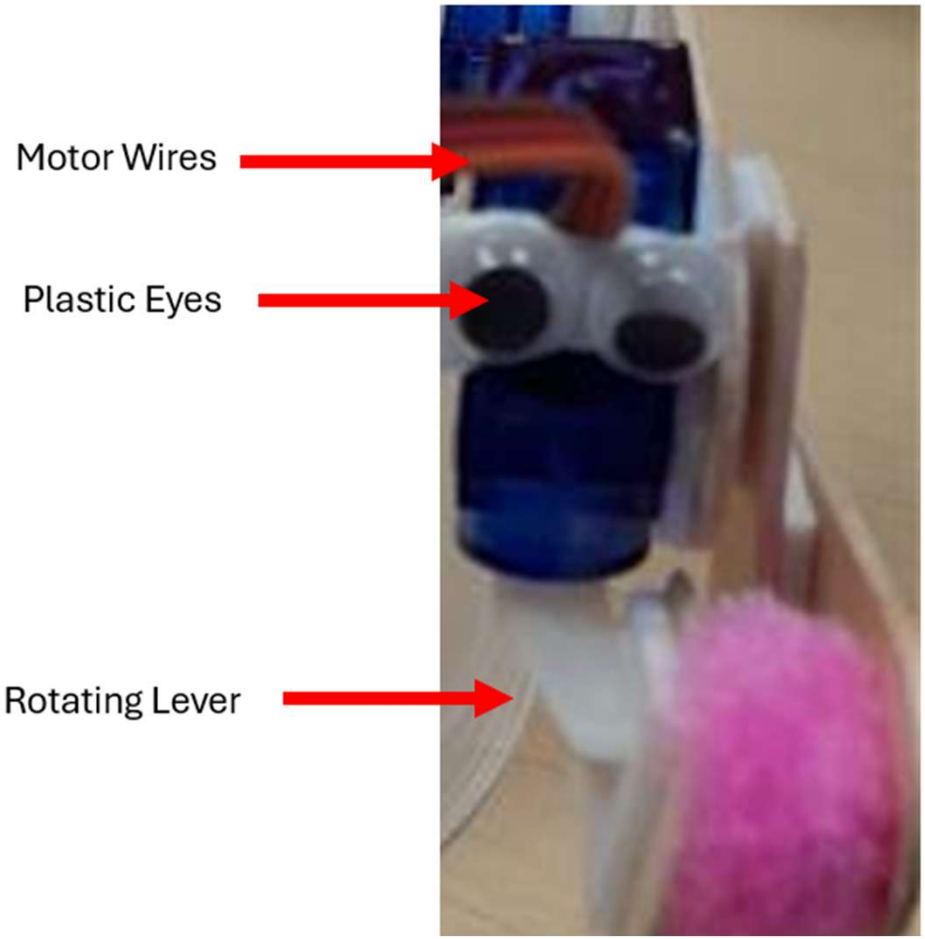
Core Components:

1. Programming: Scratch programming to control motors
2. Electronics: Arduino Microcontroller, wiring
3. Mechanics: Base, Arm, Grabber, Motors


Arduino
Ice Cream Stick/Arm
USB Cable
Arm Motor
Male-Male Jumper Cables
Gripper Motor
Tub

Build the Robotic Gripper:

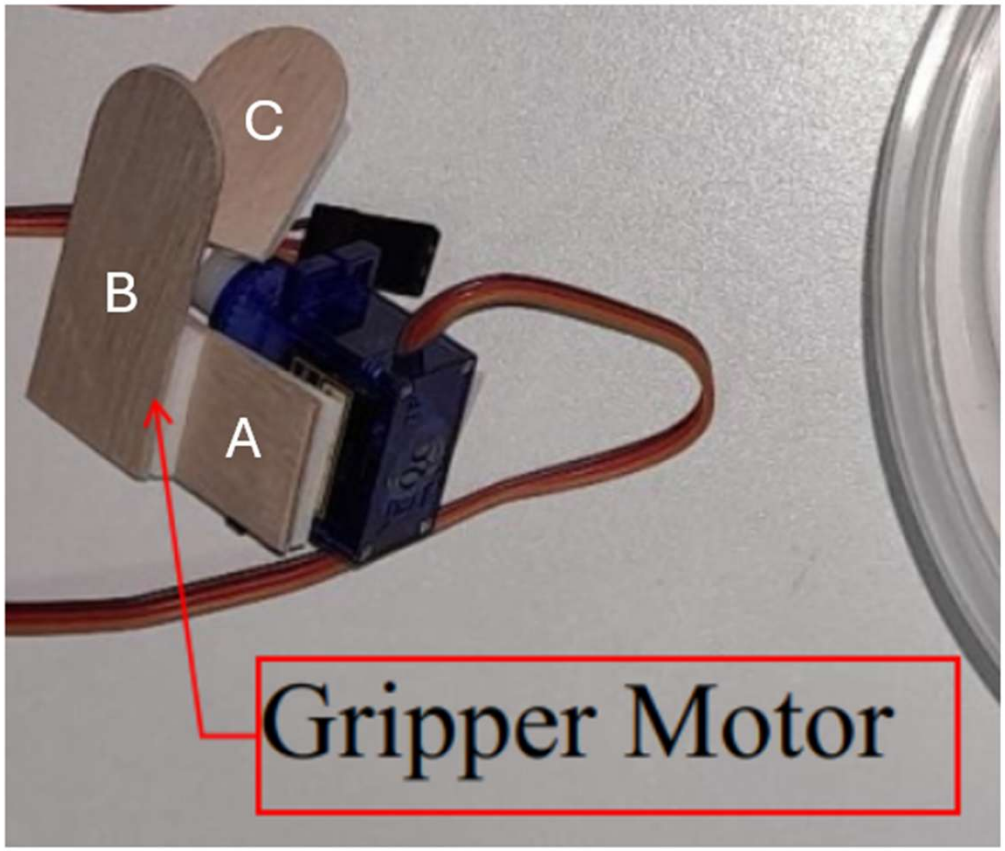
On your motor you have a section where the wires are coming out. This is the front of your motor. Then there is a white plastic lever at the bottom as shown here:



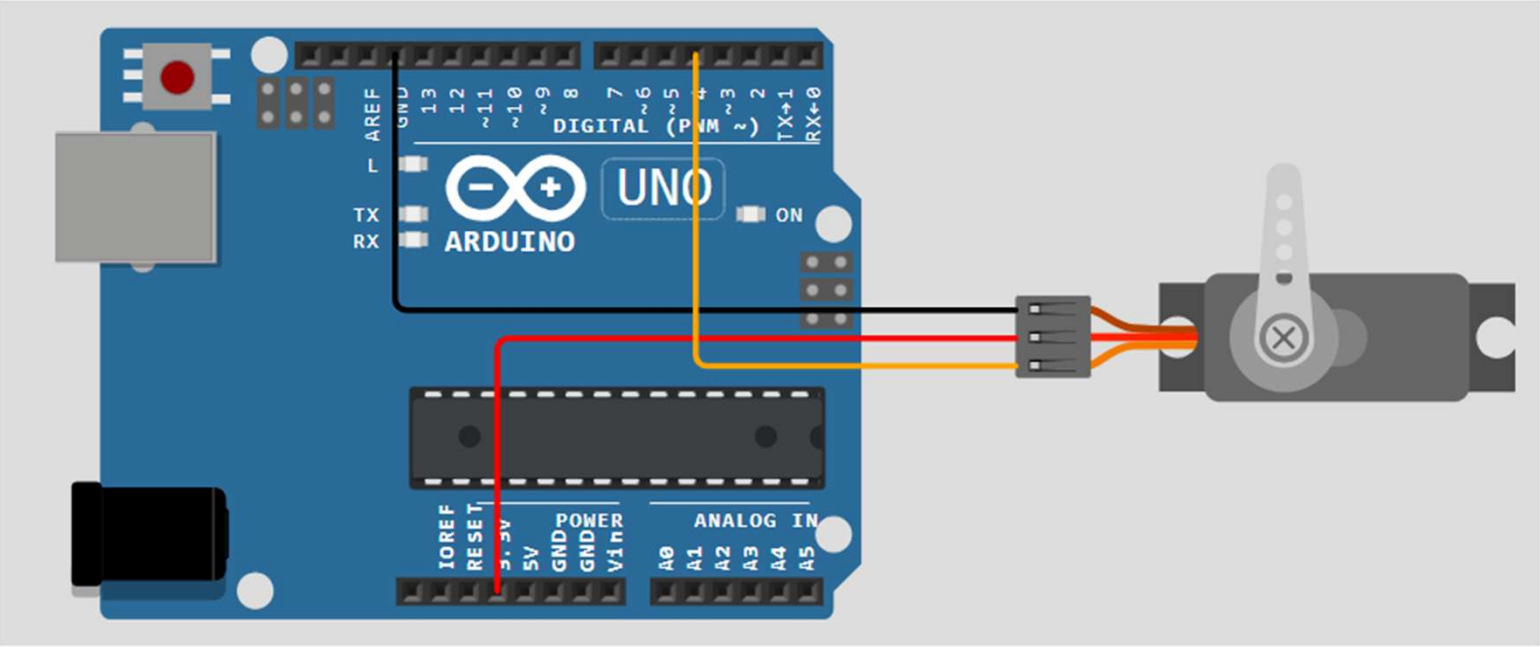
Take the two plastic eyes in your tub and a small piece of double sided tape and secure it just below where the wires come out.

This will help you remember where the front of the motor is when it turns.

Now, also in your tub you have 3 small parts, A, B, C for the gripper motor, shown here:



- Connect a GND (ground) pin from your Arduino to the brown connector on your Gripper motor
- Connect pin 4 from your Arduino to the orange connector on your Gripper motor



Build the Robotic Arm:

Planning:

Before constructing your Robotic Arm first get familiar with your items and how best to secure them together:

- Flip your tub around, it will act as your base for your Robotic arm
- Your Arduino will sit on the bottom of the tub
- Your lifting servo motor should sit on side of the tub base – where is a good spot?
- Your ice cream stick arm needs to be secured to the lifting servo motor – where is a good spot?
- At the end of ice cream stick arm you need to attach your grabber servo motor – where is a good spot?

Using double sided tape:

- Secure your Arduino to the tub:

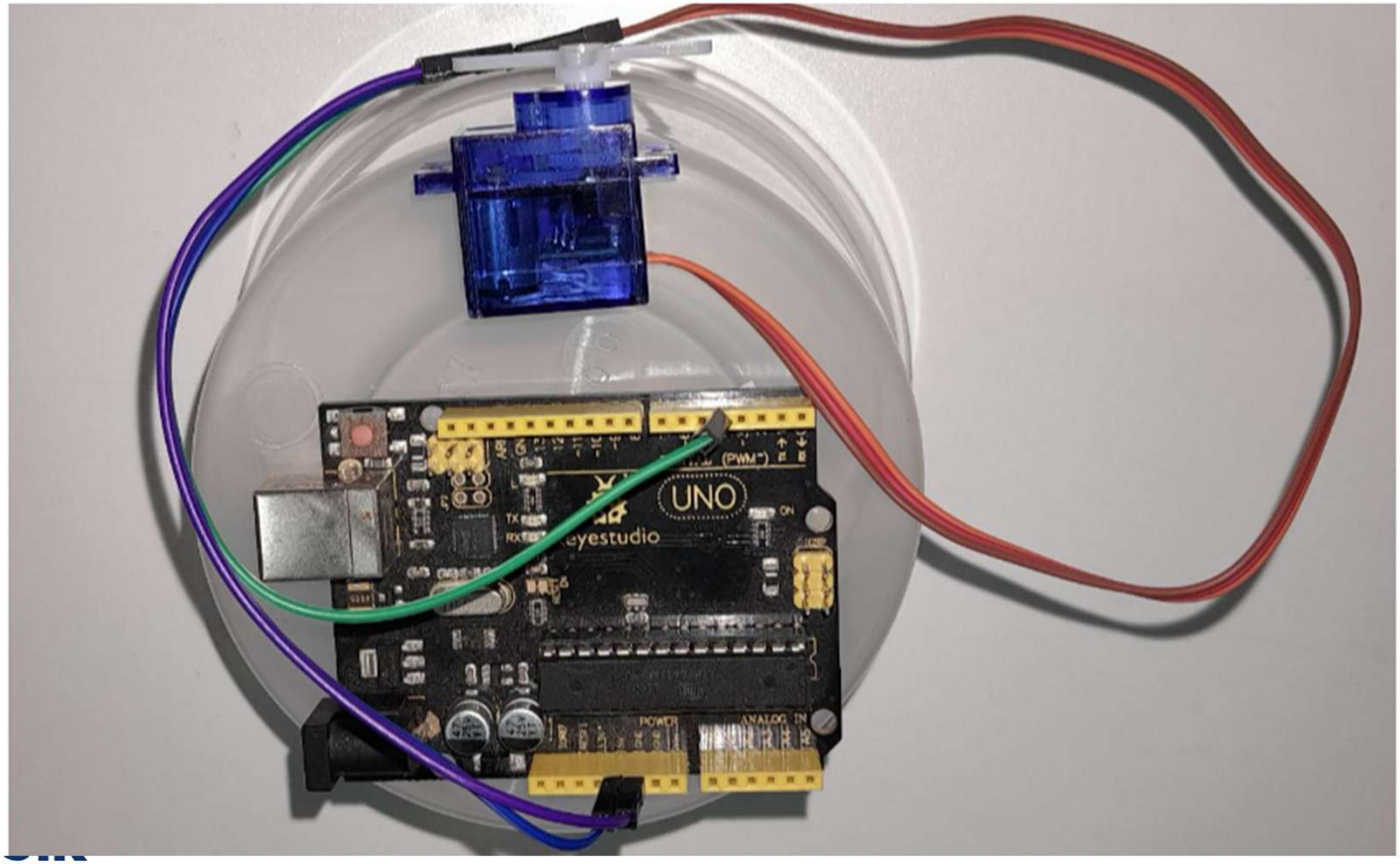
Using double sided tape:

- Secure your Arduino to the tub:

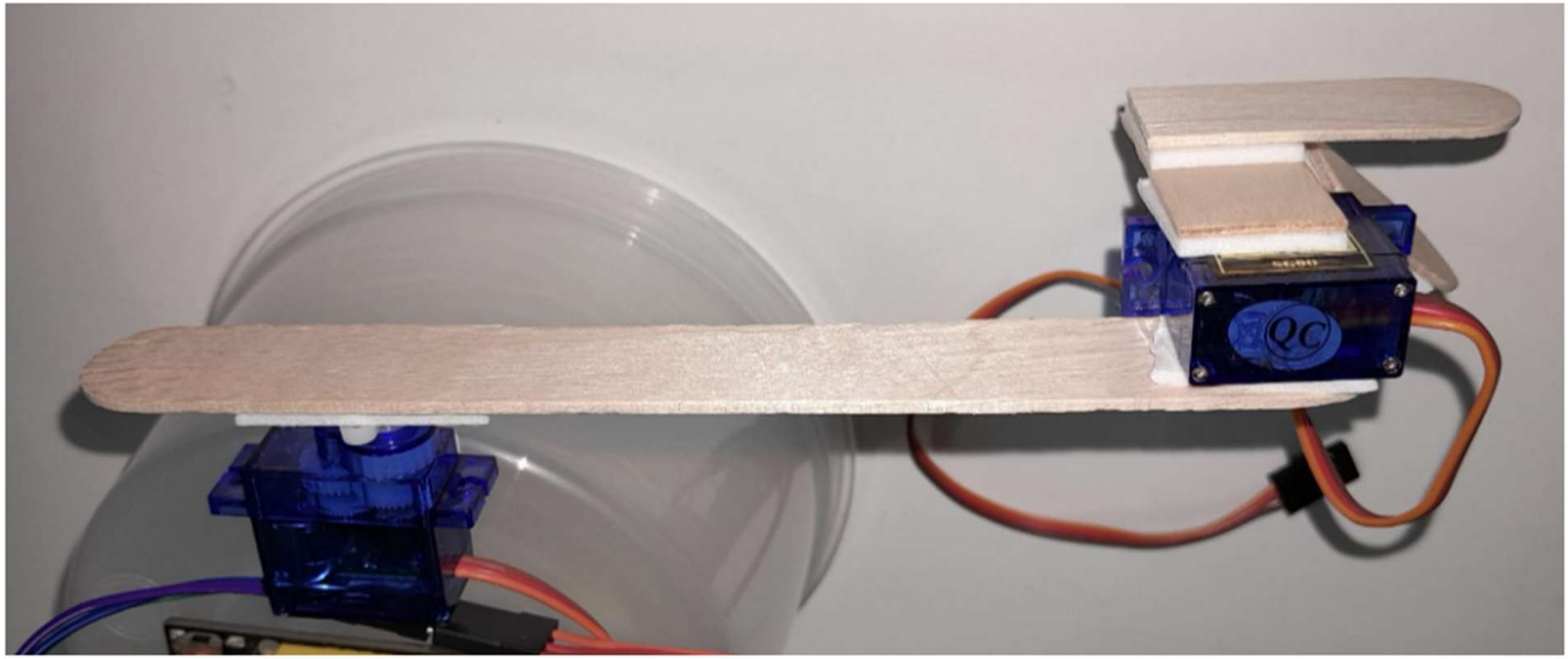
- Next attach the Arm motor to the base and connect the ice cream stick to the Arm motor:

Using the male to male wires:

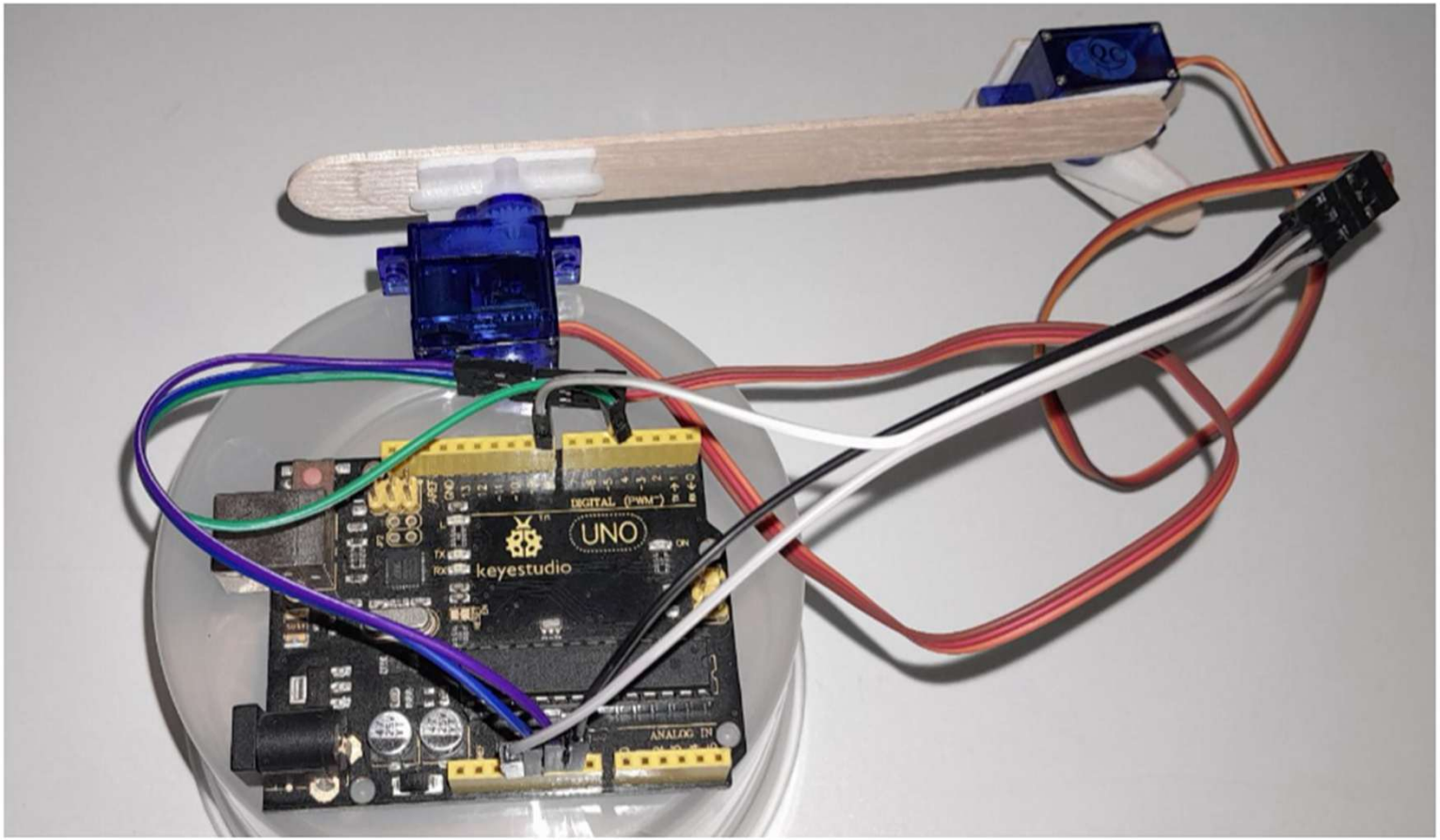
- Connect the 5V pin from your Arduino to the red connector on your Arm motor
- Connect a GND (ground) pin from your Arduino to the brown connector on your Arm motor
- Connect pin 8 from your Arduino to the orange connector on your Arm motor



- Attach the the Gripper motor to the end of Arm motor stick:

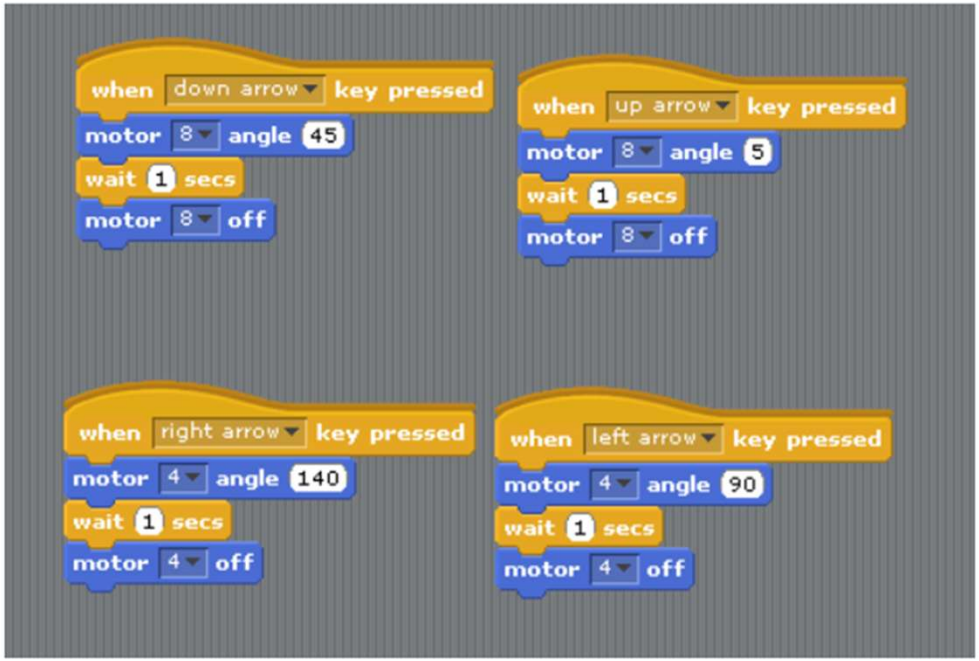


- Next connect the wires if not already connected:



Testing the Robotic Arm:

Build a new project using the following Scratch code blocks:



However, these angles that were chose may not work for you as each Robotic arm is unique!

You need to use trial and error to figure out what are the best angles to do the lifting and grabbing!

Your final goal is to try to lift up the fluff ball

Reflections

What resources limitations did you have?

What are the limitations of the Robotic Arm, what can and can it not do?

What alternative designs can you recommend or different materials used?