EE105: Introduction to Electrical Engineering

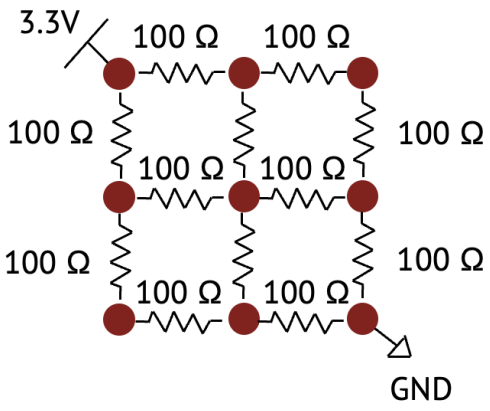
Asymmetric Touchpanel

Objective:

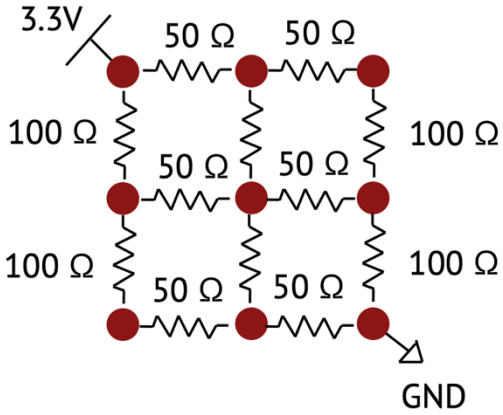
In this lab, we'll learn the basics of Asymmetric Touchpanel, how it works and how it can provide better touch location than the Symmetric Touchpanel. We will also cover analog voltage readout using Arduino and how it can be integrated with sensors.

Symmetric vs Asymmetric Touchpanel:

Due to the asymmetry in the resistance values, the asymmetric touchpanel provides different voltages at each node of the circuit.

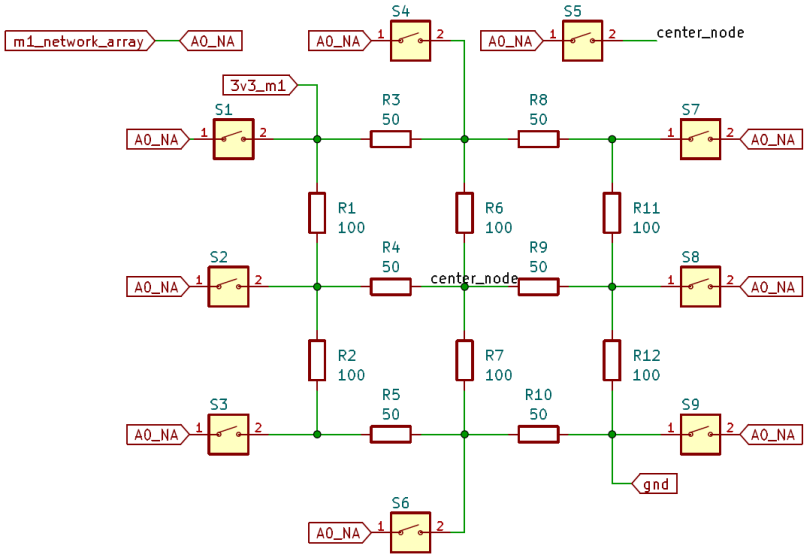


Symmetric Touchpanel



Asymmetric Touchpanel

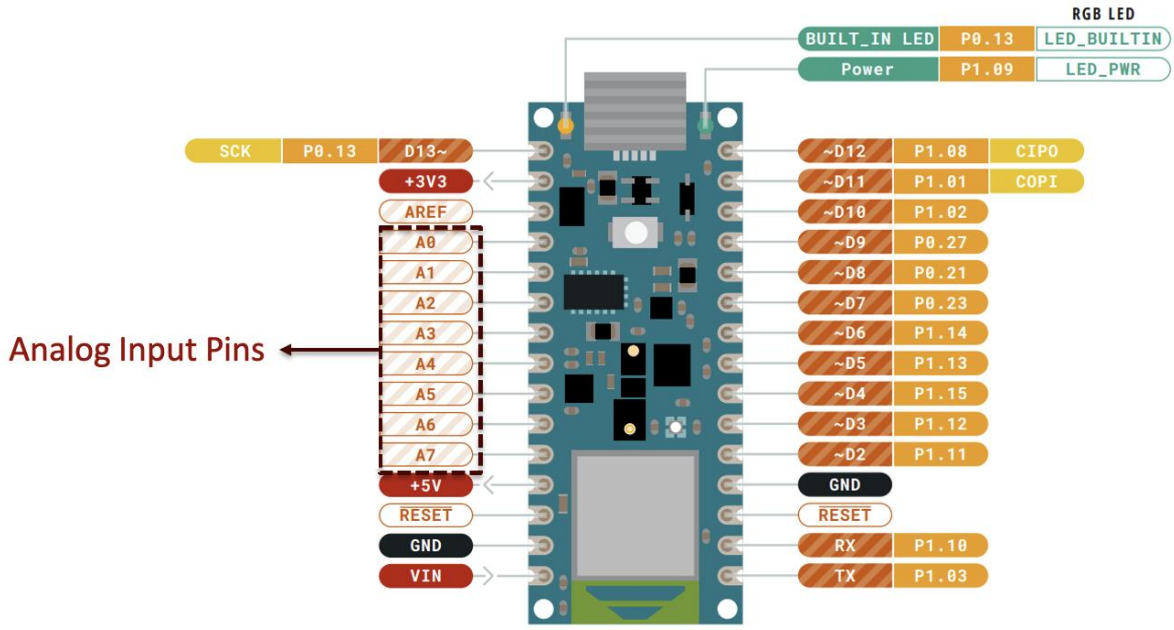
Network-1 of the Dev. Board:



The network-1 of module-1 has switches connected to the analog pin (A0) of the Arduino. When a switch is pressed, the A0 pin gets connected to that corresponding node, allowing us to measure voltage of that node.

Checkpoint-1: Find selector J7 in your dev board and connect the center pin to Module-1 pin.

Analog voltage readout using Arduino:



Pinout of Arduino Nano 33 BLE

Arduinos are inherently digital systems. However, they are equipped with analog input pins which can measure analog voltages and convert them to digital signals with the help of built-in ADCs (Analog to Digital Converters).

Checkpoint-2: Open your Arduino IDE and go to *File* > *Examples* > *Basics* > *ReadAnalogVoltage*.

A sketch will open which will read analog voltage and print the voltage in the serial monitor. The example code should look like this:

```
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5 / 1023.0);
  // print out the value you read:
  Serial.println(voltage);
}
```

Checkpoint-3: Find out the reference voltage and bit count of ADCs your Arduino boards. (A simple internet search should be good enough for now).

Checkpoint-4: Make necessary changes to the example code based on the reference voltage and the ADC resolution (bit count of ADC) of your system. This code should be able to detect voltage from the analog pin A0 of Arduino.

Detect which button has been pressed:

The following function should allow you to find which button has been pressed based on the detected voltage on the Arduino.

```
void m1_network_array(){
  Serial.println("Make sure that module 1 is active");
  while(1){
    int analog_value = analogRead(m1_na);
    if(analog_value > 1000 && analog_value < 1020) Serial.println("S1 (3.3V)");
    else if (analog_value > 590 && analog_value < 615)Serial.println("S2 (1.95V)");
    else if (analog_value > 360 && analog_value < 375)Serial.println("S3 (1.18V)");
    else if (analog_value > 755 && analog_value < 775)Serial.println("S4 (2.50V)");
    else if (analog_value > 500 && analog_value < 520)Serial.println("S5 (1.65V)");
    else if (analog_value > 240 && analog_value < 260)Serial.println("S6 (0.804V)");
    else if (analog_value > 635 && analog_value < 660)Serial.println("S7 (2.12V)");
    else if (analog_value > 400 && analog_value < 425)Serial.println("S8 (1.35V)");
    else if (analog_value > 0 && analog_value < 200)Serial.println("S9 (0V)");
    delay(20);
  }
}
```

Checkpoint-5: Use this function to verify that your system finds the correct button whenever it is pressed. If your code doesn't work, compare with the code from checkpoint-2