

# HSHL HACKATHON 2025



HOCHSCHULE  
HAMM-LIPPSTADT

## Time table

- **19.03.2025 Wednesday 12:00-19:00 -**
- First day of the event
- Getting to know the hardware
- **20.03.2025 Thursday 10:30-19:00 -**
- Starting with the tasks
- **21.03.2025 Friday 10:30-17:00 -**
- Finishing the tasks
- 17:00-18:00
- Counting of the points.
- Announcement of the winner team.

# THE BOX

- Each team is given one **BOX** of components listed further and on **THE BOX** itself
- The Hackathon competition is comprised of **5 Tasks**
- You have enough parts in **THE BOX** to complete any task, but not every task at the same time.
- **Plan accordingly.**

# THE BOX

## parts

**THE BOX** contains parts, for which you are **responsible as a team:**

- Arduino Uno Microcontroller x2
- Breadboard x1
- Numpad x1
- Servo motor 9g x3
- Servo Motor 360° x1
- Button x2
- Photoresistor x3
- USB Wire For Arduino x2
- RGB Diode x1
- Piezo Speaker x1
- IR sensor x3
- Display x1
- Variable resistor x3
- Radio Module x2



# THE BOX parts



# Regarding the Tools and **THE BOX**

**Your team is responsible for the equipment you have.**

**At the end of competition, we expect to get the box and the tools as they were given out.**

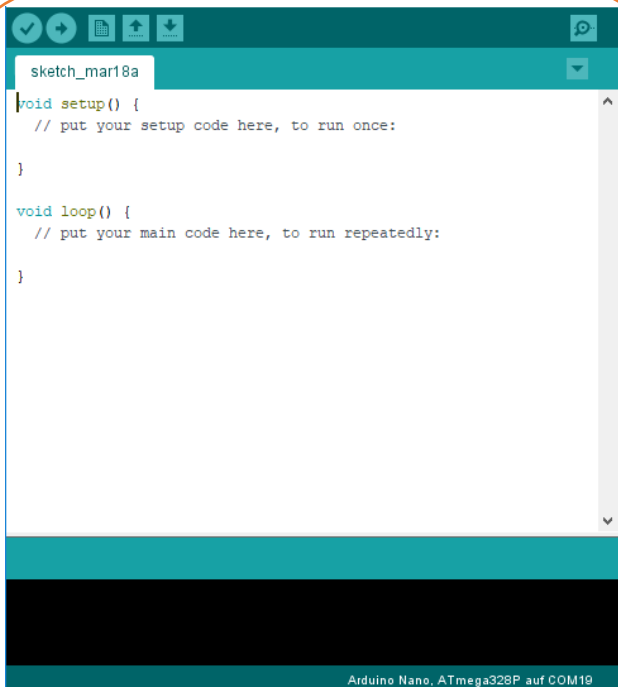
- Egregious material hogging
- malicious destruction of parts, tools and materials
- negligent destruction of parts, tools and materials
- repeated destruction of parts, tools and materials

## **May result in**

- stern talking's to,
- points deduction,
- long and uncomfortable gazes,
- walks of shame,
- expulsions,
- heavy fines,
- etc.

**Jokes aside, just be careful and immediately speak to one of the organisers if something is bended, broken, glued shut, fuming or generally **not performing like you would expect it to.****

# Arduino Uno



```
sketch_mar18a
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

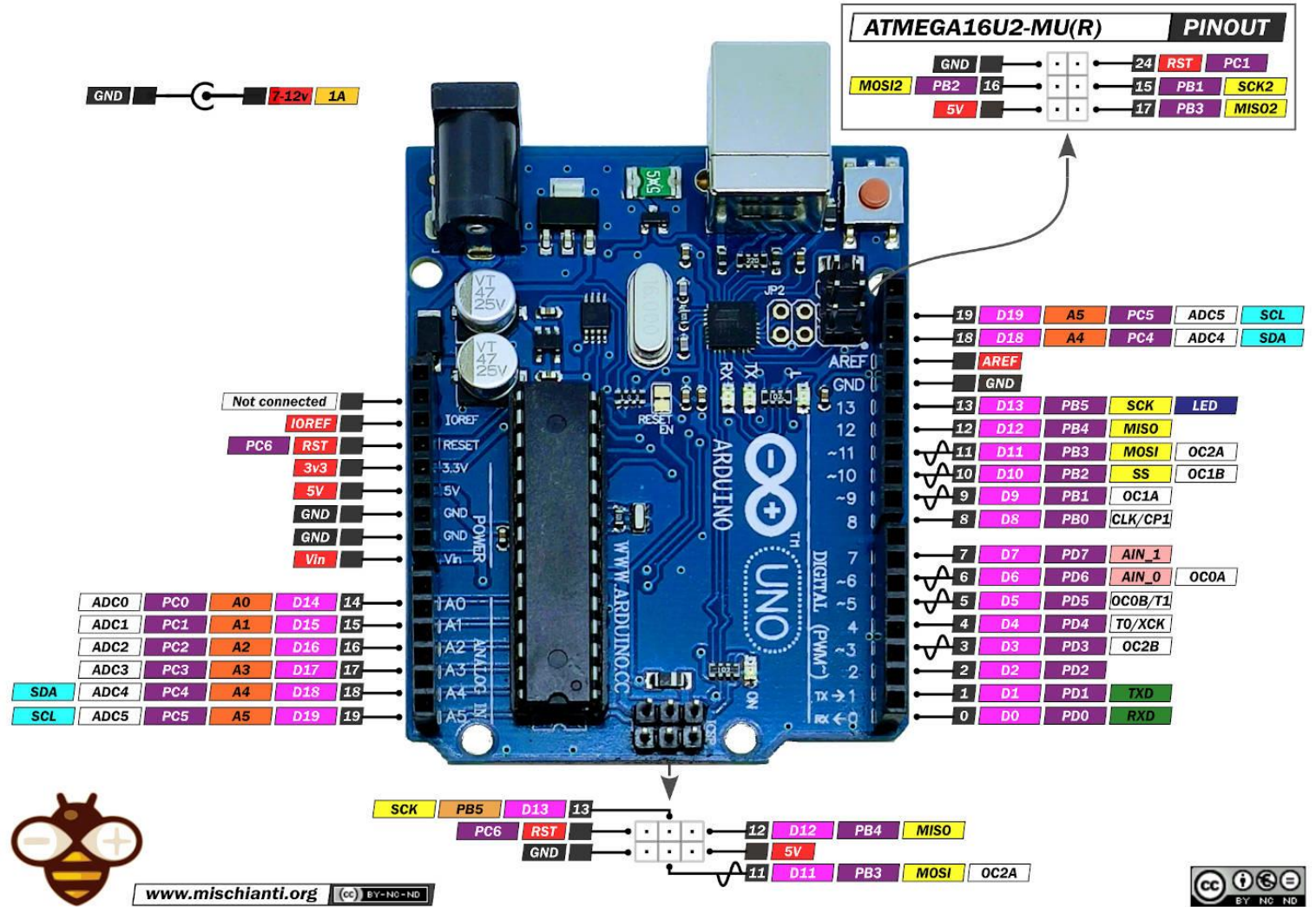
Arduino Nano, ATmega328P auf COM19

- **You know it**
- **You love it**
- **Open Arduino IDE**
- **Put what you need for setup in setup**
- **Put what you need in a loop in a loop**
- **Connect sensors or motors to the various pins**
- **Connect your Arduino with a usb**
- **Choose the right port and press Load  
→ And it will just do it!**

# Arduino Uno

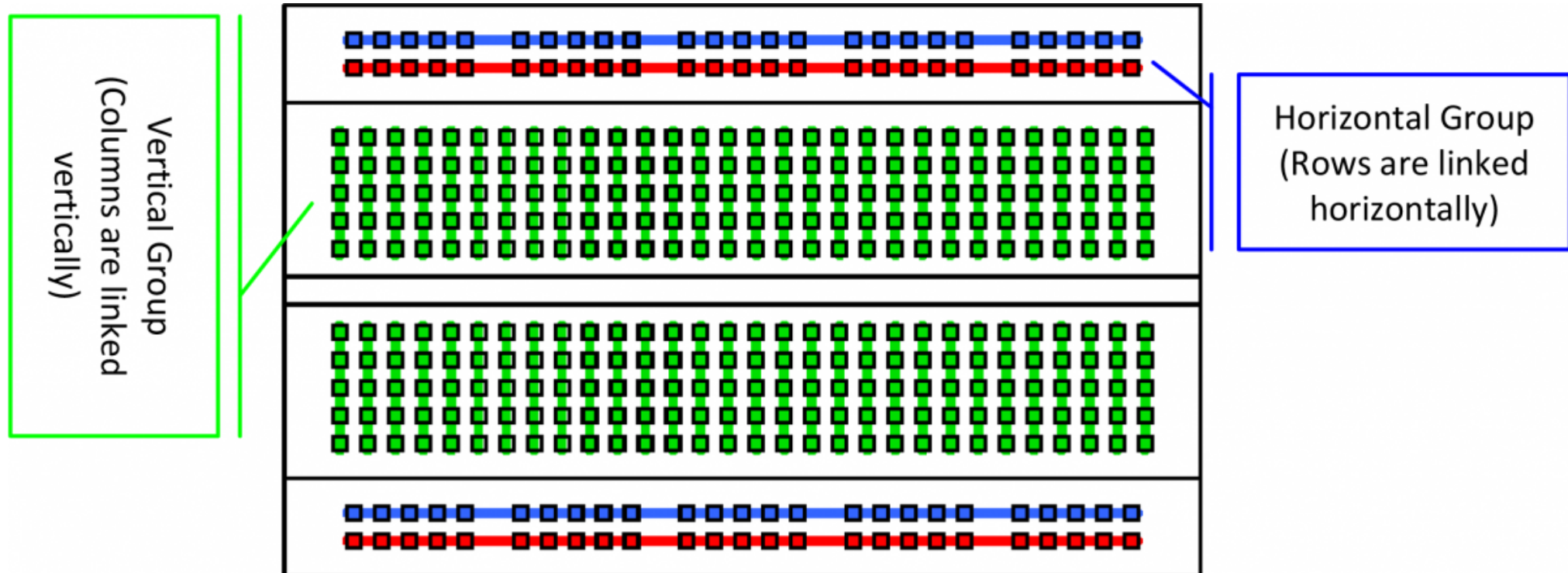
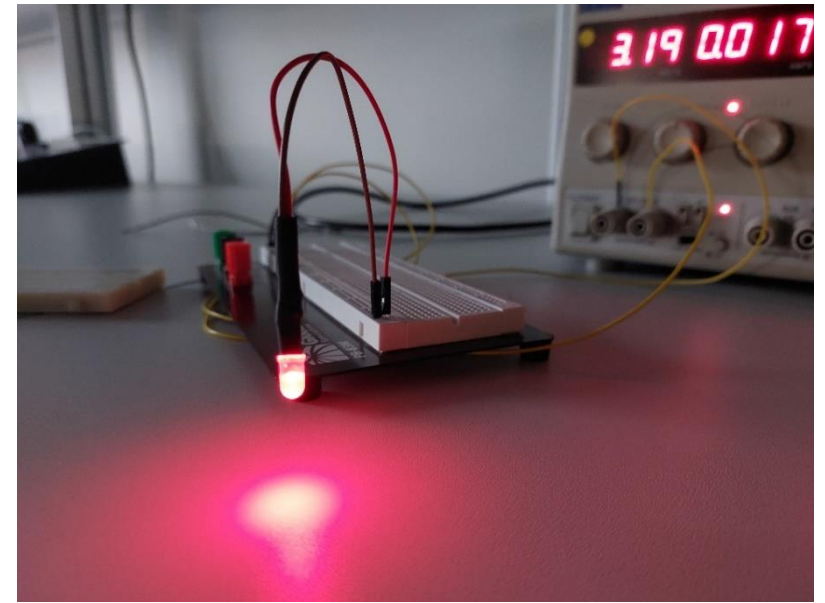
## Arduino UNO Rev3

## PINOUT



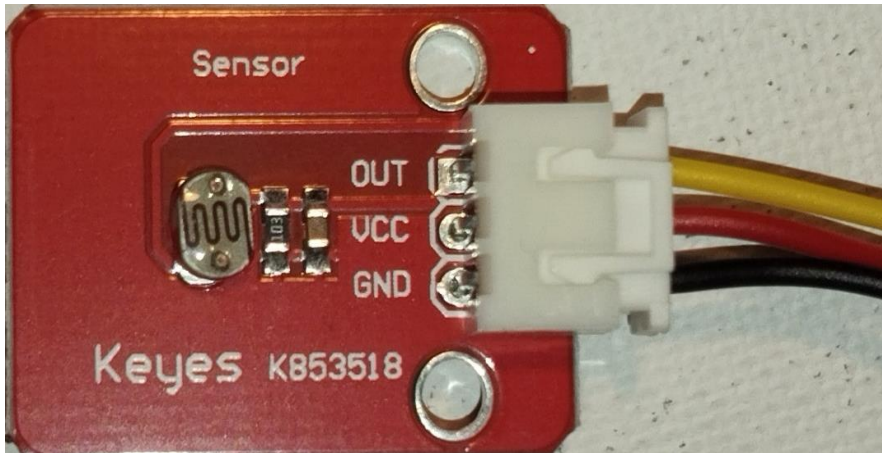


# Breadboard



# Photoresistor

Photoresistor Module – Detects light levels using a photoresistor (LDR - Light Dependent Resistor). **Resistance decreases with more light, changing the voltage, which the Arduino reads as an analog value.**



## Wiring:

OUT – YELLOW – A0-A5  
VCC – RED – +5V  
GND – BLACK – GROUND

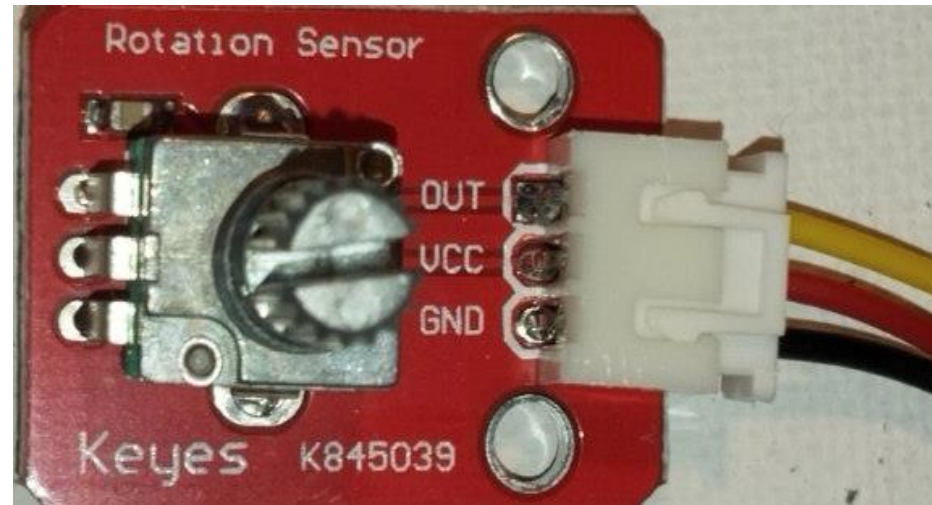
## Coding:

```
1 void setup() {  
2     Serial.begin(9600);  
3     //Init A5 to INPUT mode  
4     pinMode(A5, INPUT);  
5 }  
6  
7 void loop() {  
8     //Write input value to COM-port  
9     Serial.println(analogRead(A5));  
10    delay(100);  
11 }
```

## Parsing:

analogRead(A5) → 0 – 1023  
Higher brightness → higher value

# Variable resistor



## Wiring:

OUT – YELLOW – A0-A5

VCC – RED – +5V

GND – BLACK – GROUND

Variable Resistor Module – A variable resistor with a rotating knob. **Turning it changes resistance, altering the output voltage, which the Arduino reads as an analog value.**

## Parsing:

`analogRead(A5)` → 0 – 1023

extreme counterclockwise position → 0

extreme clockwise position → 1023

## Coding:

```
1 void setup() {  
2     Serial.begin(9600);  
3     //Init A5 to INPUT mode  
4     pinMode(A5, INPUT);  
5 }  
6  
7 void loop() {  
8     //Write input value to COM-port  
9     Serial.println(analogRead(A5));  
10    delay(100);  
11 }
```

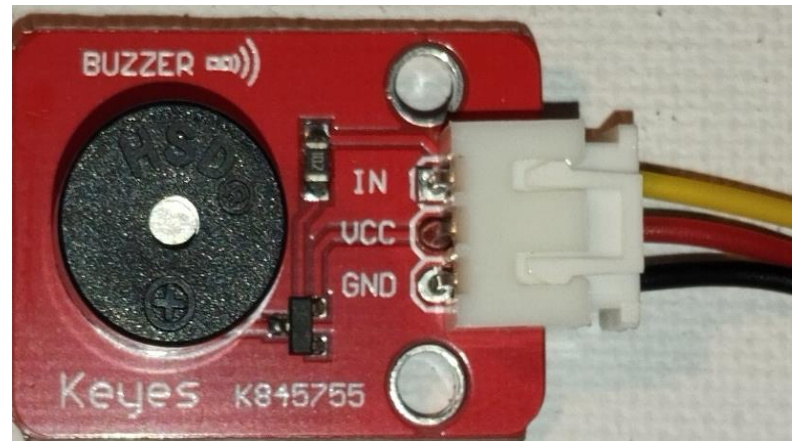
# Buzzer

Active Buzzer Module – **Produces sound when powered.**

It has a built-in oscillator, so it **beeps with just a HIGH signal from the Arduino.**

## Parsing:

`digitalWrite(2, HIGH)` → turn buzzer on  
`digitalWrite(2, LOW)` → turn buzzer off



## Wiring:

IN – YELLOW – D0-D13

VCC – RED – +5V

GND – BLACK – GROUND

## Coding:

```
1 void setup() {  
2     //Init D2 to OUTPUT mode  
3     pinMode(2, OUTPUT);  
4 }  
5  
6 void loop() {  
7     //Turn buzzer on  
8     digitalWrite(2, HIGH);  
9     delay(500);  
10    //Turn buzzer off  
11    digitalWrite(2, LOW);  
12    delay(500);  
13 }
```



# Button

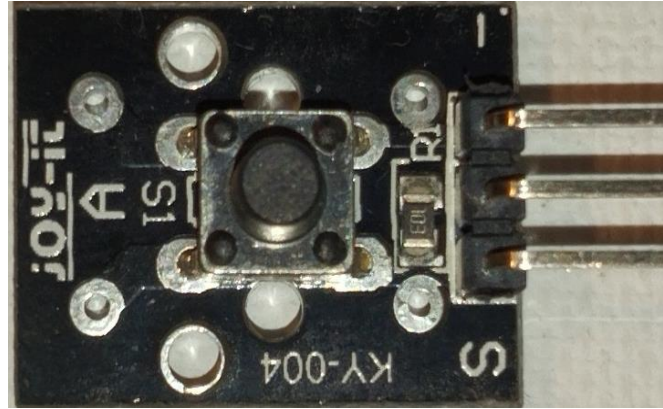
Button Module – A push-button with a built-in pull-up resistor. **When pressed**, it connects to GND, making **the output LOW**. **When released**, **the output is HIGH**.

## Parsing:

`digitalRead(2)` → 0 or 1

Button pressed → 0

Button not pressed → 1



## Wiring:

- – GROUND
- middle – +5V
- S** – SIGNAL – D0-D13

## Coding:

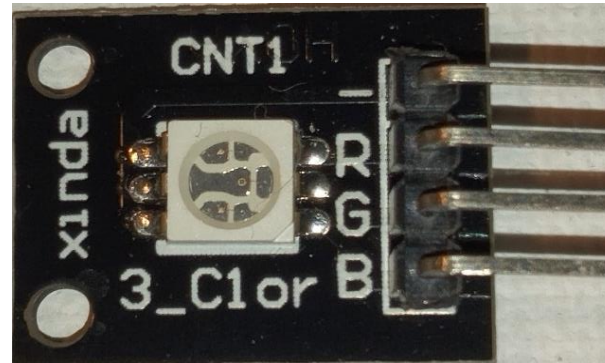
```
1 void setup() {  
2     Serial.begin(9600);  
3     //Init D2 to INPUT mode  
4     pinMode(2, INPUT);  
5 }  
6  
7 void loop() {  
8     //Write input value to COM-port  
9     Serial.println(digitalRead(2));  
10    delay(100);  
11 }  
12
```

# RGB LED

RGB LED Module – Contains Red, Green, and Blue LEDs in one package. **Adjusting PWM signals on each color pin mixes different colors.**

## Parsing:

digitalWrite(2, HIGH) → turn on LED on pin 2  
digitalWrite(2, LOW) → turn off LED on pin 2



## Wiring:

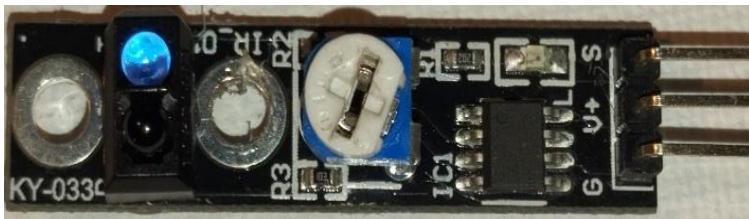
- GROUND
- R** – RED – D0-D13
- G** – GREEN – D0-D13
- B** – BLUE – D0-D13

## Coding:

```
1 void setup() {
2     //Init D2, D3 and D4 to OUTPUT mode
3     pinMode(2, OUTPUT);
4     pinMode(3, OUTPUT);
5     pinMode(4, OUTPUT);
6 }
7
8 void loop() {
9     //Shining one LED at a time
10    digitalWrite(4, LOW);
11    digitalWrite(2, HIGH);
12    delay(1000);
13    digitalWrite(3, HIGH);
14    digitalWrite(2, LOW);
15    delay(1000);
16    digitalWrite(4, HIGH);
17    digitalWrite(3, LOW);
18    delay(1000);
19 }
```

# IR sensor

Line Tracking Sensor (Digital) – Uses an infrared sensor to detect dark or light surfaces. Outputs LOW on white (built-in LED ON) and HIGH on black (built-in LED OFF).



## Wiring:

S	– SIGNAL – D0-D13
V+	– +5V
G	– GROUND

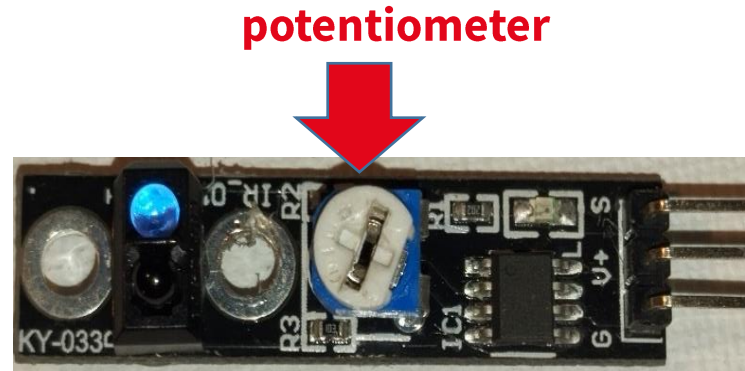
## Coding:

```
1 void setup() {  
2     Serial.begin(9600);  
3     //Init D2 to INPUT mode  
4     pinMode(2, INPUT);  
5 }  
6  
7 void loop() {  
8     //Write input value to COM-port  
9     Serial.println(digitalRead(2));  
10    delay(100);  
11 }  
12
```

## Parsing:

digitalRead(2) → 0 or 1  
Surface is white → 0 (LED is ON)  
Surface is black → 1 (LED is OFF)

# IR sensor



## Wiring:

S	– SIGNAL – D0-D13
V+	– +5V
G	– GROUND

## Sensitivity adjustment:

For reading custom barcodes with the **Line Tracking Sensor**, adjust the sensitivity using the **potentiometer** so it correctly differentiates between black and white areas. Turn:

- **Clockwise** → Increases sensitivity (detects lighter marks as black).
- **Counterclockwise** → Decreases sensitivity (requires darker marks to trigger).

Fine-tune it until the sensor reliably detects patterns.



# Servo motor (9g)

A 9g servo motor is a small and lightweight actuator that **rotates within a fixed range (typically 0-180°)**.

It is controlled using PWM signals and is widely used in robotics, RC planes, and small automation projects.



## Wiring:

YELLOW – D0-D13 (PWM)  
RED – +5V  
BLACK – GROUND

## Tuning:

Replaceable caps

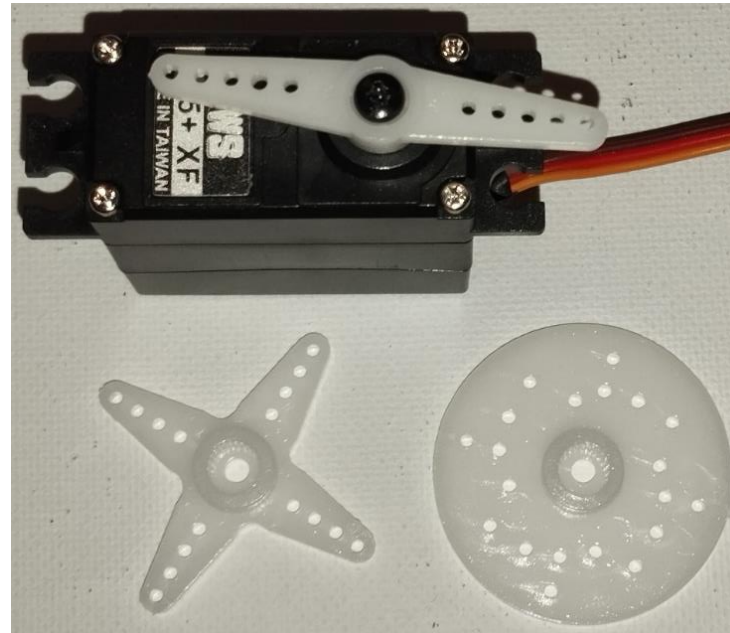
# Servo motor (9g)

```
1 Use library:
2 #include <Servo.h>
3 Basic functions:
4 Servo myservo; // create Servo object to control a servo
5 myservo.attach(9); // attaches the servo on pin 9 to the Servo object
6 myservo.write(pos); // tell servo to go to position in variable 'pos'
7 //Keep in mind:
8 //Servo needs time to move to the desired position. The usual way //to do that is:
9 int old_pos = 80;
10 int new_pos = 180;
11 ✓ for (pos = old_pos; pos <= new_pos; pos += 1) {
12     myservo.write(pos); // tell servo to go to position in variable 'pos'
13     delay(15); // waits 15 ms for the servo to reach the position
14 }
```

# Servo motor (360°)

A continuous rotation servo is a modified **servo motor that can rotate 360° in either direction** instead of stopping at a fixed angle.

The speed and direction are controlled by PWM signals.



## Wiring:

YELLOW – D0-D13 (PWM)  
RED – +5V  
BLACK – GROUND

## Tuning:

Replaceable caps

## Coding:

```
1 Use library:
2 #include <Servo.h>
3 Basic functions:
4 Servo myservo; // create Servo object to control a servo
5 myservo.attach(9);
6 //Attaches the servo to pin 9
7 myservo.write(pos);
8 //Tell the servo to go with speed in variable 'pos'
9 //”pos” controls the speed and direction of the
10 //servo, with 90 as no rotation/no movement, and 0 and 180
11 //being extremes in ether directions
```

# Display



A 16x2 LCD with I<sup>2</sup>C is a display module that **shows 16 characters per row on two rows**. The I<sup>2</sup>C interface reduces the number of required pins (only SDA and SCL) compared to the standard parallel connection.



I<sup>2</sup>C address of the display is printed on the back

## Wiring:

G	– GROUND
VCC	– +5V
SDA	– SDA (A4)
SCL	– SCL (A5)

## Coding:

```
1 Use library:
2 #include <LiquidCrystal_I2C.h>
3 Basic functions:
4 LiquidCrystal_I2C lcd(0x27, 16, 2);
5 //Set the LCD I2C address to
6 //0x27 for a 16 chars and 2-line display
7 lcd.init(); // initialize the lcd
8 lcd.backlight();
9 lcd.noBacklight();
10 lcd.setCursor(col, row);
11 lcd.print(data); //print data as a string
12 lcd.clear();
```



# Keypad

The PmodKYPD is a 4x4 matrix keypad with 16 buttons, commonly used for user input in microcontroller projects. It connects using row-column scanning.



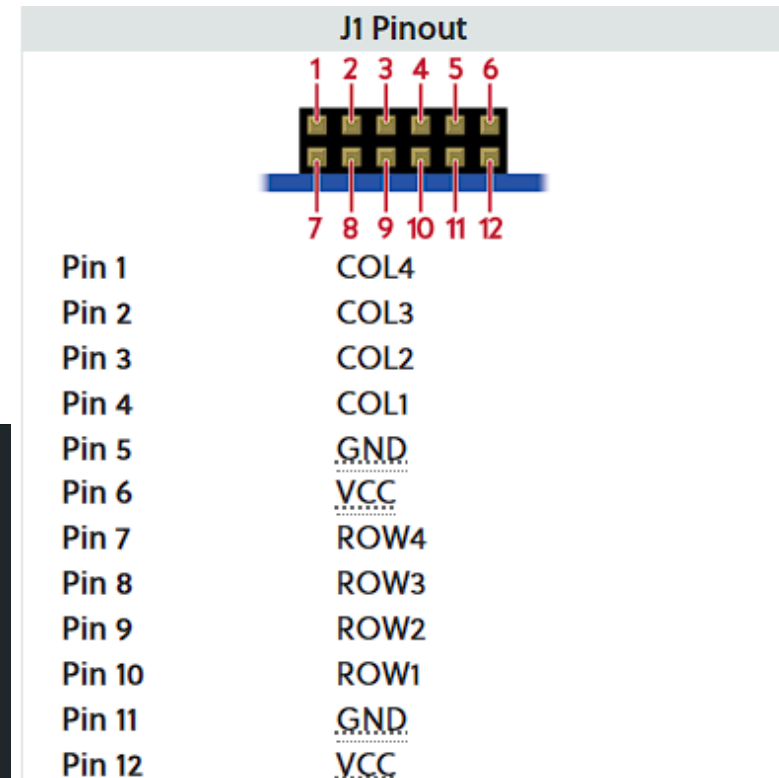
# Keypad

## Coding:

```

1  #include <KYPD.h>
2  KYPD myKYPD; // create KYPD object to control a keypad
3  unsigned int col[4] = { col1, col2, col3, col4 };
4  unsigned int row[4] = { row1, row2, row3, row4 };
5  myKYPD.setPins(row, col); //set the pins
6  int keyTable[4][4] = { { 49, 52, 55, 48 }, //Define keymap to the
7                          { 50, 53, 56, 70 }, //values on the PmodKYPD
8                          { 51, 54, 57, 69 }, //(This table contain
9                          { 65, 66, 67, 68 } }; //corresponding ASCII
10 myKYPD.setKeyMap(keyTable); //codes. You can copy-paste it)
11 myKYPD.begin();
12 myKYPD.end(); //OPTIONAL!! Free pins to do other things
13 int key = myKYPD.getKey(); //Returns -1 if no key pressed, otherwise
14 //returns corresponding value from keyTable.
15 //Therefore, after checking for -1 key can be converted to char:
16 if (key != -1)
17     Serial.println((char)key);

```



## Wiring:

VCC – +5V  
GND – GROUND  
other pins – D0-D13

# Radio module

A **radio module** enables wireless data transmission using radio waves. It consists of a **transmitter** and **receiver** for communication between devices.

## APC220

- **UHF (418–455MHz), UART-based** communication.
- **Up to 1 km range, simple serial interface.**

## nRF24L01

- **2.4GHz transceiver, SPI-based** communication.
- **Short-range, fast data transfer, supports multiple devices.**



# Radio module

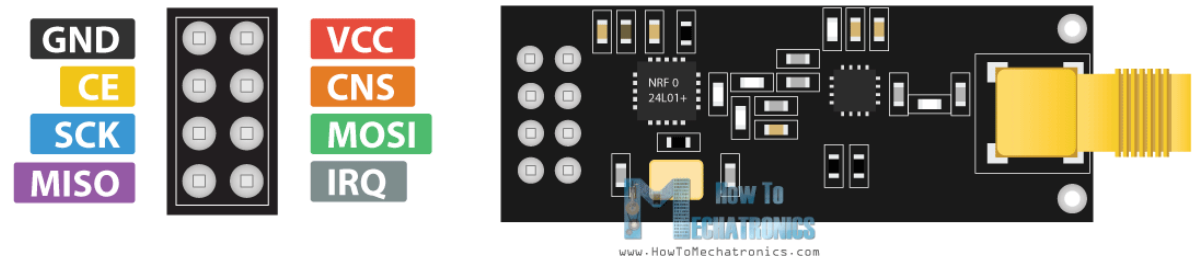
## Wiring:

GND	– GROUND
VCC	– +3.3V
MOSI	– D11
MISO	– D12
SCK	– D13
CE	– D0-D10
CNS	– D0-D10
IRQ	– not connected

### NRF24L01 Pinout



### NRF24L01+ PA/LNA Pinout

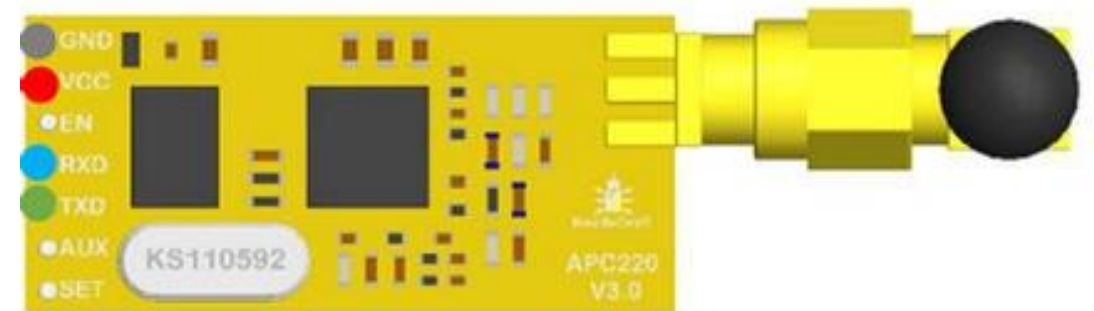




# Radio module

## Wiring:

<b>GND</b>	– GROUND
<b>VCC</b>	– +5V
<b>RX</b>	– D2-D13
<b>TX</b>	– D2-D13
<b>SET</b>	– D2-D13
<b>EN</b>	– not connected
<b>AUX</b>	– not connected





# Warm up tasks

- The competition will start tomorrow, before which you will need to familiarise yourself with the equipment at hand.
- For this we prepared a number of warm ups

- Make a numpad connected to arduino input numbers into IDE console using serial connection.
- Make an Arduino application that gives a number 0 to 9 every second to console depending on the knob position of variable resistor.
- Print “Hello world” to display.
- Send “Hello world” on radio module. Receive it.
- Make a blue servo do a 180°, 90°, back and forth.
- Make a 360° servo move. Now move it back.
- When photo resistor reads light, make the piezo speaker scream.
- See what IR sensor gives using serial. Point it at stuff.
- Lay your equipment into **THE BOX** carefully and present to jury for completion for practice. You can do it multiple times. Be our guest.

