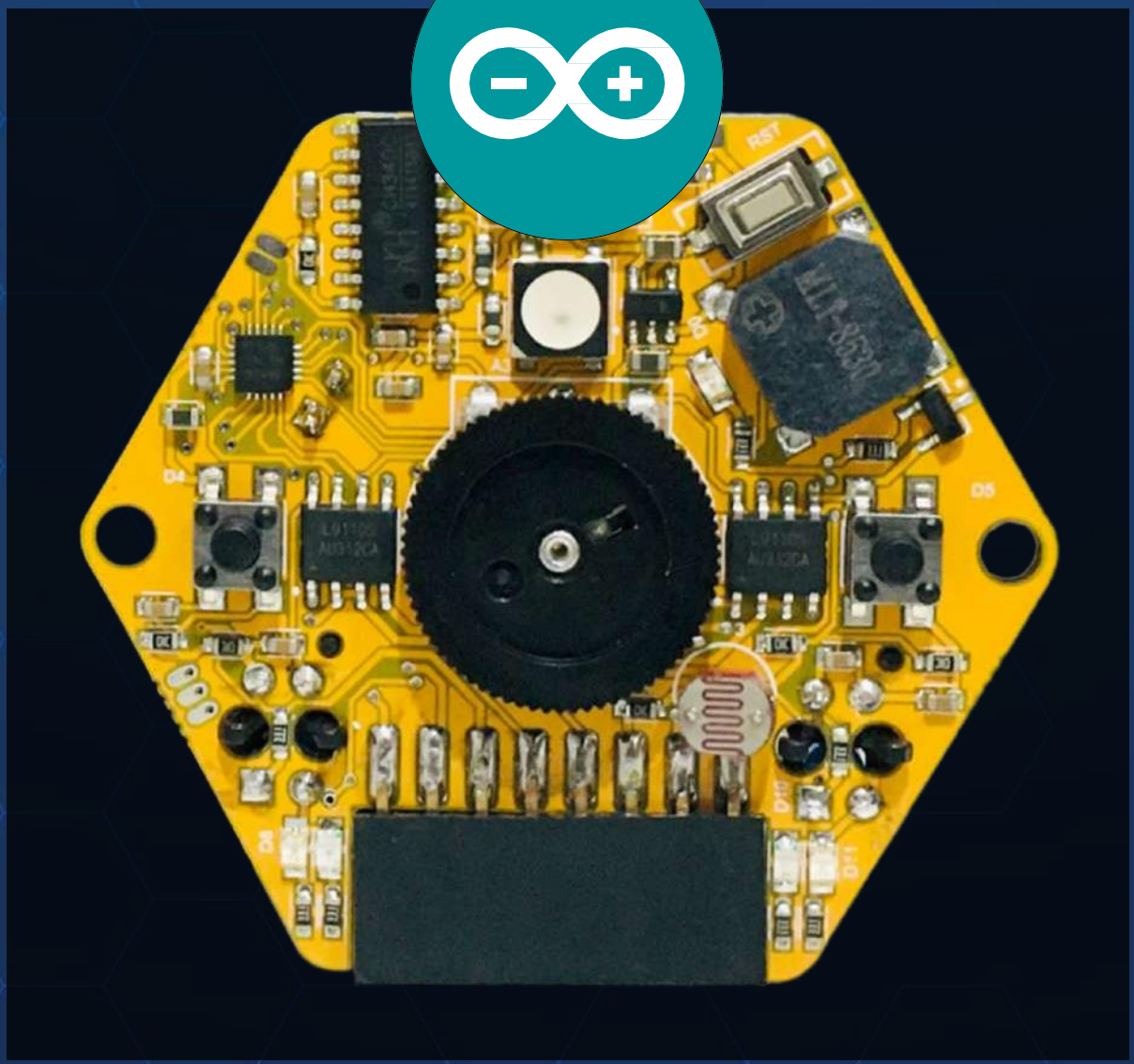
magicbit

TINY

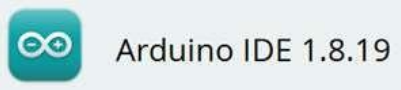
WITH ARDUINO



All the lessons in this course are centered around the Arduino programming language, providing a comprehensive and hands-on approach to learning. By engaging with Arduino, you will gain practical experience in coding, electronics, and hardware integration, fostering a deep understanding of programming concepts and real-world applications.

Note: Installing and adding Magicbit Tiny to Arduino IDE is previously done at the OS updating step. If you haven't updated the OS before, follow the below steps.

Installing Arduino Software

- Download one of the below versions of Arduino software to your computer from the below link. Select the correct option according your computer OS type (Windows , Mac or Linux)
  - Version 1.8.19 

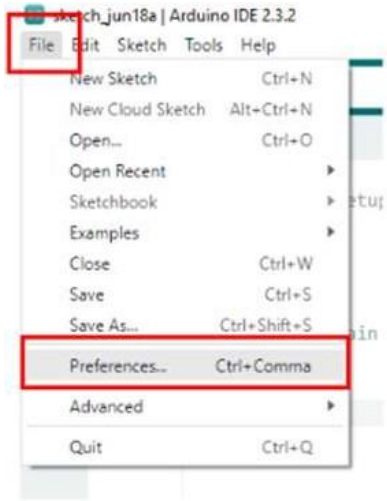
  - Version 2.3.2 Note :- Magicbit Tiny doesn't support Arduino latest versions (2.3.3 and 2.3.4)
  - Download Arduino from here - https://www.arduino.cc/en/software
- Install the downloaded software in your computer.
- Open the Arduino IDE from the shortcut created on your computer desktop.

Adding the Magicbit Tiny board to Arduino IDE

- Connect the Magicbit Tiny board to the computer using the USB cable.

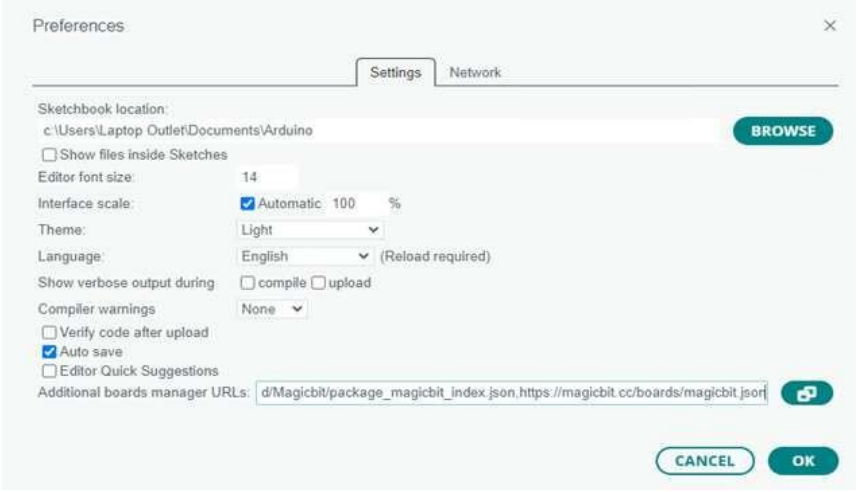
- Open the **"Preferences"** window of the Arduino IDE as follows.

File > Preferences



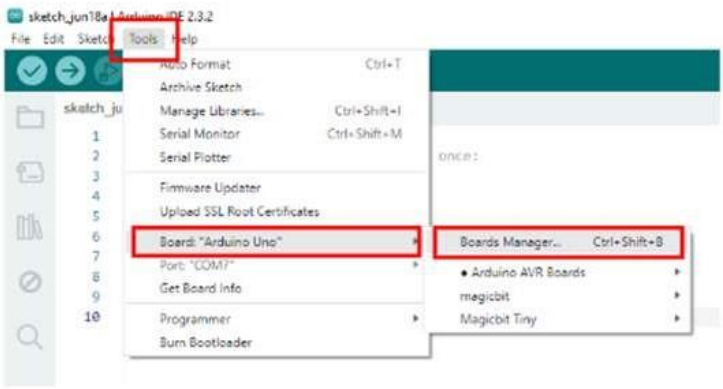
- Copy the below given release link and paste it on the **"Additional Board Manager"**. Here you can paste multiple release links by separating them by " , ".

Release Link for Magicbit Tiny: https://magicbit.cc/boards/magicbit.json

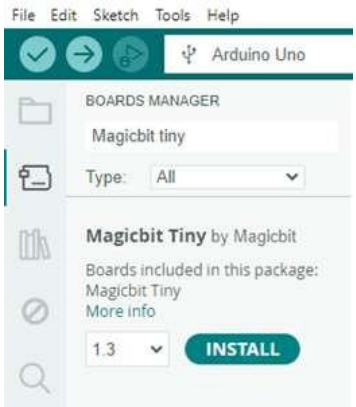


- Now open the **"Boards Manger"** window as in below path.

Tools > Board > Boards Manager



- Search as **"Magicbit Tiny"** and click on the **"INSTALL"** button to install it. This will take few minutes.

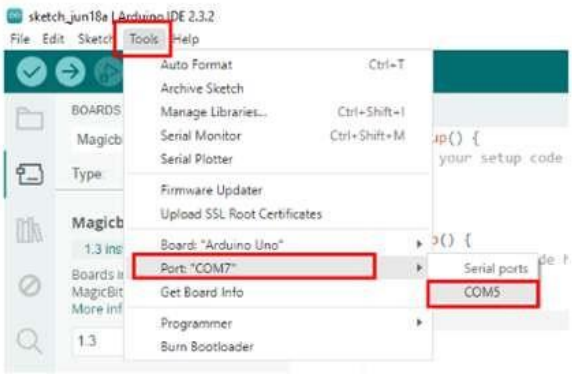


- After installing, connect with the Magicbit Tiny board as follows.

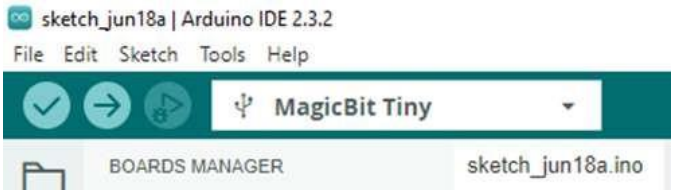
Tools > Boards > Magicbit Tiny > Magicbit Tiny

- Then select the relevant USB port which the Magicbit Tiny is connected.

Tools > Port > COM



- Now you are ready to program the Magicbit Tiny.



Note :- If you don't get the COM port to select or it doesn't appear, You need to update the drivers of your computer.

- Get into Magicbit Web Uploader and download the drivers file in **"Install Drivers"** tab.
- Install the downloaded driver and check for the COM port.
- Unplug the USB cable and also reload the Arduino IDE.

Add uploader image and highlight Install drivers tab

01. Light Show

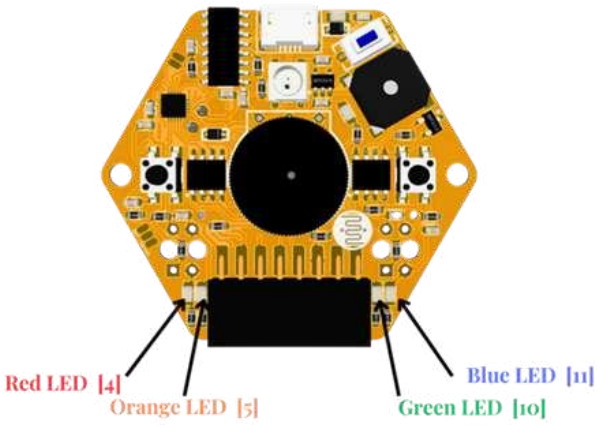
Create a LED Light Pattern using the on-board LEDs

✦ Expected Output

- https://youtu.be/J8UhgEjXZqg

✦ Description

- The LED light pattern project is a captivating display of sequential LED lighting using the Magicbit Tiny Board. With a simple array of digital output pins and a defined time delay, this project creates an eye-catching back-and-forth movement of LEDs. The speed and pattern of the LED chaser can be easily customized for various visual effects.



- In Magicbit tiny has four in-built LEDs that can be programmed. These four LEDs have four different colors.

LED Color	Pin Number
Red	4
Orange	5
Green	10
Blue	11

STEP 01 Define the pins of the LEDs as an array

```
int pinArray[] = { 4, 5, 10, 11 };
```

STEP 02 Define the speed of the pattern

The speed of the LED chaser is controlled by the `**timer**` variable, allowing users to adjust the delay between each step and customize the visual effect.

```
int timer = 100; // Speed of LED pattern can be customized here
```

- Increase the timer to decrease the speed.
- Decrease the timer value to increase the speed.

STEP 03 Configure Magicbit tiny pin numbers 4, 5, 10, and 11 as outputs

```
void setup() {
  // Configure Magicbit tiny pin numbers 4, 5, 10, and 11 as outputs
  for (int i = 0; i < 4; i++) {
    pinMode(pinArray[i], OUTPUT);
  }
}
```

STEP 04 Forward and Backward LED chaser light patterns

```
void loop() {
  // Forward LED chaser light pattern
  for (int i = 0; i < 4; i++) {
    digitalWrite(pinArray[i], HIGH);
    delay(timer);
    digitalWrite(pinArray[i], LOW);
    delay(timer);
  }

  // Backward LED chaser light pattern
  for (int i = 3; i >= 0; i--) {
    digitalWrite(pinArray[i], HIGH);
    delay(timer);
    digitalWrite(pinArray[i], LOW);
    delay(timer);
  }
}
```

Full Program

[Download the program code from here]

STEP 05 Follow the steps below to upload example code for project light pattern.

1. Connect Magicbit Tiny using the USB cable to a computer.
2. Open the Arduino IDE and get a New Sketch from the **File menu.**
3. Then in the **Tool** tab, select **Board -> Magicbit Tiny -> MagicBit Tiny**
4. In the same **Tool** tab select **Port -> COM<your port number>**
5. Type or copy and paste the above Arduino code to your new sketch and upload the code.

Once uploaded, the Magicbit Tiny Board will execute the LED light pattern, creating a mesmerizing back-and-forth movement of lights.

Experiment with different LED configurations, change the order of pin connections, or adjust the `**timer**` variable to create unique and visually appealing LED chaser effects.

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/sE5F5A8FOQA

02. Dimmer Magic

Create a program to control the brightness of the LEDs using the Potentiometer in the Magicbit Tiny.

✦ Expected Output

- https://youtu.be/LWfj0KK7BGE

✦ Description

- The LED Dimmer project allows us to control the brightness of LEDs using a potentiometer. By turning the potentiometer, we can increase or decrease the intensity of the light emitted by the LEDs.

✦ How it works

- **Potentiometer Input:** The potentiometer, connected to pin A3 on the Magicbit Tiny board, acts as our brightness controller. Rotating the potentiometer wheel generates different analog values.

- **LED Brightness Control:** The analog value from the potentiometer is mapped to a range of 0 to 255. This mapped value represents the brightness level.

- **LED Pins Configuration:** LEDs are connected to digital pins 4, 5, 10, and 11. These pins will receive the analog value and adjust the brightness of the connected LEDs accordingly.

Steps for the Program

STEP 01 Define the component pins

```
#define POT_PIN A3

// Define an array of output pins
int pinArray[] = { 4, 5, 10, 11 };
```

STEP 02 Configure the pins

```
void setup() {
  // Configure pins 4, 5, 10, and 11 as outputs
  for (int count = 0; count < 4; count++) {
    pinMode(pinArray[count], OUTPUT);
  }
}
```

STEP 03 Get the potentiometer reading, map the values and update the pattern

```
void loop() {
  // Read the analog value from pin A3 (potentiometer)
  int potValue = analogRead(POT_PIN);

  // Map the analog value to a range of 0 to 255
  int mappedValue = map(potValue, 0, 1023, 0, 255);

  // Update the analog output of each pin in pinArray with the mapped value
  for (int i = 0; i < 4; i++) {
    analogWrite(pinArray[i], mappedValue);
  }
}
```

Full Program

[Download the program code from here]

STEP 04 Follow the steps below to upload example code for project dimmer magic.

1. Connect Magicbit Tiny using the USB cable to a computer.
2. Open the Arduino IDE and get a New Sketch from the File menu.
3. then in the Tool tab, select Board -> Magicbit Tiny -> MagicBit Tiny
4. In the same Tool tab select port -> COM<your port number>
5. Type or copy and paste above Arduino code to your new sketch and upload the code

After uploading the code.

1. Rotate the potentiometer wheel clockwise to increase brightness.
2. Rotate the potentiometer wheel counterclockwise to decrease brightness.

Learning Points:

- Analog Input: Understanding analog values from the potentiometer.
- Mapping Values: Learn how to map analog values to a specific range (0 to 255).

Syntax

- map(value, fromLow, fromHigh, toLow, toHigh)

Parameters

- value: the number to map.
- fromLow: the lower bound of the value's current range.
- fromHigh: the upper bound of the value's current range.
- toLow: the lower bound of the value's target range.
- toHigh: the upper bound of the value's target range

PWM Output: Explore the concept of Pulse Width Modulation (PWM) for LED brightness control.

After a call to analogWrite(), the pin will generate a steady rectangular wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite()) on the same pin.

Try to make diffrent light paterns by controlling the LED brightness.
Have fun exploring the world of LEDs and brightness control with the LED Dimmer project!

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/dr9HBz4XL8g

03. Melody Magic

ACTIVITY 03

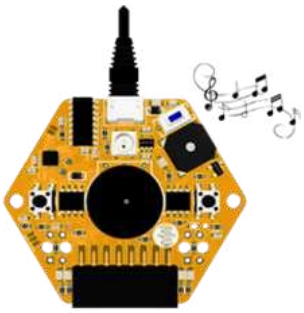
Create a program to play musical tones using the buzzer in the Magicbit Tiny.

Expected Output

- https://youtu.be/emDmRRFRifw

Description

- Welcome to the magical world of melodies with the Magicbit Tiny Melody project! This fun and interactive project will introduce you to the basics of creating musical tunes using the Magicbit Tiny board.
- Here to play a simple tune using the in-built buzzer. By providing a list of musical notes and their corresponding frequencies, we can create delightful melodies.
- Let's dive into the simple documentation and exploration of melodies.



How it works

1. Musical Notes:

- We have defined musical notes from C4 to D8, along with their frequencies.
- Each note is represented by its frequency in Hertz.

2. Melody Array:

- We've created a melody array containing notes and their durations.
- The code plays each note in sequence to produce a melody.

3. Buzzer Output:

- The in-built buzzer in the Magicbit Tiny board is used to generate sound.
- The `**tone()**` function plays each note, and `**noTone()**` stops the sound.

Steps for the Program

STEP 01 Define musical notes and corresponding frequencies

```
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
#define REST      0
```

STEP 02 Define the component pins

```
int buzzer = 13;
```

STEP 03 Define the melody using above defined notes

```
int melody[] = {
  NOTE_C4, 4, NOTE_C4, 8,
  NOTE_D4, -4, NOTE_C4, -4, NOTE_F4, -4,
  NOTE_E4, -2, NOTE_C4, 4, NOTE_C4, 8,
  NOTE_D4, -4, NOTE_C4, -4, NOTE_G4, -4,
  NOTE_F4, -2, NOTE_C4, 4, NOTE_C4, 8,

  NOTE_C5, -4, NOTE_A4, -4, NOTE_F4, -4,
  NOTE_E4, -4, NOTE_D4, -4, NOTE_AS4, 4, NOTE_AS4, 8,
  NOTE_A4, -4, NOTE_F4, -4, NOTE_G4, -4,
  NOTE_F4, -2,
};

// Calculate the number of notes
int notes = sizeof(melody) / sizeof(melody[0]) / 2;

// Calculate the duration of a whole note in milliseconds
int wholenote = (60000 * 4) / 180;

// Variables to store the note duration and divider
int divider = 0, noteDuration = 0;
```

STEP 04 Define the setup

```
void setup() {
  // Initialize the buzzer pin
  pinMode(buzzer, OUTPUT);

  // Iterate over the notes of the melody
  for (int thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {
    // Calculate the duration of each note
    divider = melody[thisNote + 1];

    if (divider > 0) {
      // Regular note, just proceed
      noteDuration = (wholenote) / divider;
    } else if (divider < 0) {
      // Dotted notes are represented with negative durations
      noteDuration = (wholenote) / abs(divider);
      noteDuration *= 1.5;  // Increase the duration in half for dotted notes
    }
    // Play the note for 90% of the duration, leaving 10% as a pause
    tone(buzzer, melody[thisNote], noteDuration * 0.9);
    // Wait for the specified duration before playing the next note
    delay(noteDuration);
    // Stop the waveform generation before the next note
    noTone(buzzer);
  }
}
void loop() {
}
```

Full Program

Download the program code from here

STEP 05 Follow the steps below to upload example code for project dimmer magic.

1. Connect Magicbit Tiny using the USB cable to a computer.
2. Open the Arduino IDE and get a New Sketch from the File menu.
3. Then in the Tool tab, select Board -> Magicbit Tiny -> MagicBit Tiny
4. In the same Tool tab select port -> COM<your port number>
5. Type or copy and paste above Arduino code to your new sketch and upload the code

Learning Points:

- **Array Usage:** Understanding how arrays can be used to store musical notes and durations.
- **Buzzer Control:** Learn how to generate different tones using the in-built buzzer.
- **Musical Notes:** Introduction to the concept of musical notes and frequencies.

Modify the melody array to create your own tunes. Try adding or removing notes to see how it changes the melody. It's your chance to become a musical magician!

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/7TaCVeZBFQc

04. Night Light

ACTIVITY 04

Create a program to activate the LEDs according to the background light level detected by the LDR in the Magicbit Tiny.

Expected Output

- https://youtu.be/g1bGuv1lukQ

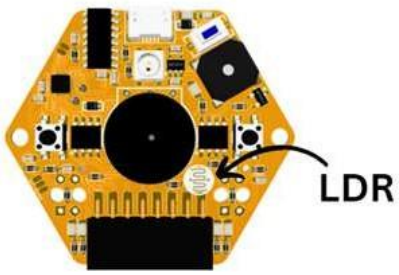
Description

- Welcome to the enchanting world of the Magicbit Tiny Night Light project! Get ready to explore how the Magicbit Tiny board can create a magical night light using its in-built Light-Dependent Resistor - LDR (Resistor that responds to the ambient light levels). As the surroundings get darker, the night light activates, illuminating the LEDs.

- Let's dive into simple documentation and uncover the secrets of this charming project.

Use components in Magicbit Tiny:

1. In-built Light-Dependent Resistor (LDR) - A7 (Analog pin 7)
2. 4 LEDs - Digital pins 4, 5, 10, 11



How it works

1. LDR Sensing:

   The Magicbit Tiny board is equipped with a Light-Dependent Resistor (LDR) that detects changes in ambient light.

2. LED Illumination:

   We've connected four LEDs to different pins (4, 5, 10, 11) on the Magicbit Tiny board.

   When the LDR senses low light (darkness), all four LEDs illuminate.

3. Control Logic:

   If the LDR value falls below a certain threshold, indicating darkness, all LEDs turn on.

   If there's enough ambient light, the LEDs turn off.

Steps for the Program

STEP 01 Define the components pins

```
const int ldrPin = A7;
const int ledPin1 = 4;
const int ledPin2 = 5;
const int ledPin3 = 10;
const int ledPin4 = 11;
```

STEP 02 Set up the pin modes

```
void setup() {
  // Set LED pins as outputs
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(ledPin3, OUTPUT);
  pinMode(ledPin4, OUTPUT);
}
```

STEP 03 Define the process

```
void loop() {
  // Read the analog value from the LDR
  int ldrValue = analogRead(ldrPin);

  // Check if the ambient light is below a certain threshold (darkness)
  if (ldrValue < 600) {  // ldr value vary from  0 - 1024
    // Turn on all LEDs
    digitalWrite(ledPin1, HIGH);
    digitalWrite(ledPin2, HIGH);
    digitalWrite(ledPin3, HIGH);
    digitalWrite(ledPin4, HIGH);
    delay(10);  // Delay for stability
  } else {
    // Turn off all LEDs
    digitalWrite(ledPin1, LOW);
    digitalWrite(ledPin2, LOW);
    digitalWrite(ledPin3, LOW);
    digitalWrite(ledPin4, LOW);
    delay(10);  // Delay for stability
  }
}
```

Full Program

Download the program code from here

STEP 04 Follow the steps below to upload example code for project night light.

1. Connect Magicbit Tiny using the USB cable to a computer.
2. Open the Arduino IDE and get a New Sketch from the **File** menu.
3. then in the **Tool** tab, select **Board -> Magicbit Tiny -> MagicBit Tiny**
4. In the same **Tool** tab select **port -> COM<your port number>**
5. Type or copy and paste the above Arduino code to your new sketch and upload the code.

Usage:

- Place your Magicbit Tiny in a dark environment, and witness the magic as the LEDs light up, creating a delightful night light effect.

Learning Points:

- **LDR Sensing:** Understanding how Light-Dependent Resistors work to detect changes in light levels.
- **LED Control:** Learning how to control LEDs based on sensor input.
- **Ambient Light:** Exploring the concept of ambient light and its impact on the LDR.

Experiment with the LDR sensitivity by adjusting the threshold value. See how the LEDs respond to different lighting conditions in your room.

Now, let the Magicbit Tiny Night Light guide you through the night with its captivating glow!

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/W00Tfj9VBRs

05. Anti-Theft

ACTIVITY 05

Create an object security alert system using the Magicbit Tiny board

✦ Expected Output

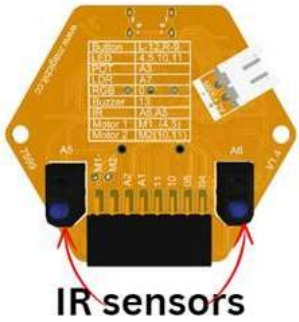
- https://youtu.be/NEFo_9V7h3s

✦ Description

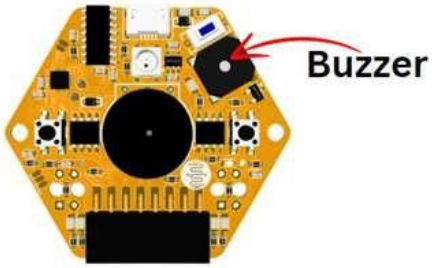
- Welcome to the exciting world of the Magicbit Tiny Anti-Theft project!
- Let's explore how the Magicbit Tiny board, equipped with two in-built Infrared (IR) sensors and a buzzer, creates a simple yet effective anti-theft mechanism.
- Get ready to understand the magic behind this fascinating project.
- The Magicbit Tiny Anti-Theft project acts as a reliable alarm mechanism, notifying you when an unauthorized person attempts to take an object placed on the Tiny IR sensors. The integration of infrared sensors and a buzzer allows for quick and easy deployment of a security solution.

✦ Use components in Magicbit Tiny:

1. In-built Infrared (IR) Sensors - A5 and A6 (Analog Pins)



2. Buzzer - Digital pin 13



✦ How it works

1. IR Sensing:

   - The Magicbit Tiny board is equipped with two in-built Infrared (IR) sensors (Left and Right).

   - These sensors detect the presence of an object placed in front of them.

2. Alarm Activation:

   - If both IR sensors detect an object (sensor values above a threshold), the alarm is activated.

3. Buzzer Control:

   - The buzzer generates a distinct sound pattern, creating an audible alert when the alarm is activated.

✦ Steps for the Program

STEP 01 Define the components pins and threshold value

```
// Define component pins
const int irSensor1Pin = A5;  // left IR
const int irSensor2Pin = A6;  // Right IR
const int buzzerPin = 13;     // buzzer

// Define IR threshold for activation
const int IR_THRESHOLD = 1015;
```

STEP 02 Make the process

```
void setup() {
}

void loop() {

  int sensor1Value = analogRead(irSensor1Pin);  //Read ir sensor value 1
  int sensor2Value = analogRead(irSensor2Pin);  //Read ir sensor value 2
  // Check whether the both IR sensors detect an object
  if (sensor1Value > IR_THRESHOLD && sensor2Value > IR_THRESHOLD) {
    activateAlarm();  // Activate the alarm if both sensors detect an object
  } else {
    deactivateAlarm();  // Deactivate the alarm if no object is detected
  }
}
  // Function to activate the alarm
  void activateAlarm() {
    // Generate a sound pattern with increasing and decreasing tones
    for (int i = 2000; i <= 3500; i += 100) {
      tone(buzzerPin, i);
      delay(20);  // Adjusted delay for better sound control
    }
    for (int i = 3500; i >= 2000; i -= 100) {
      tone(buzzerPin, i);
      delay(20);
    }
  }
  // Function to deactivate the alarm
  void deactivateAlarm() {
    // Turn off the buzzer to deactivate the alarm
    noTone(buzzerPin);
    delay(20);
  }
```

Full Program

Download the program code from here

STEP 03 Follow the steps below to upload example code for project anti-theft.

1. Connect Magicbit Tiny using the USB cable to a computer.
2. Open the Arduino IDE and get a New Sketch from the **File** menu.
3. Then in the **Tool** tab, select **Board -> Magicbit Tiny -> MagicBit Tiny**
4. In the same **Tool** tab select **port -> COM<your port number>**
5. Type or copy and paste above Arduino code to your new sketch and upload the code.

Place the object on the designated area covering both IR sensors to activate the security system.

Learning Points:

- **IR Sensing:** Understanding how Infrared (IR) sensors detect the presence of an object.
- **Alarm Activation:** Learning the logic behind activating the alarm based on sensor readings.
- **Buzzer Control:** Exploring how to generate distinct sound patterns using a buzzer.

Now, let the Magicbit Tiny Anti-Theft project guard your belongings with its magical security features!

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/LBKRhoHLkDQ

06. Tiny Buttons

ACTIVITY 06

Create a program to control the LEDs using the push buttons in the Magicbit Tiny.

✦ Expected Output

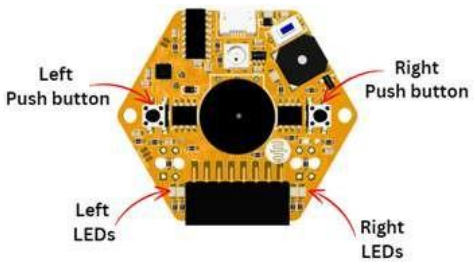
- https://youtu.be/r8PI2EycvD0

✦ Description

- Welcome to the captivating world of the Magicbit Tiny Push Button LED Control project! Let's embark on an exciting journey to learn how to control LEDs using the Magicbit Tiny board. This project introduces the concept of push buttons and demonstrates how pressing different buttons can illuminate specific sets of LEDs.
- By pressing the left button, the left set of LEDs lights up, and by pressing the right button, the right set of LEDs illuminates.

✦ Use components in Magicbit Tiny:

1. Left Push Button - Digital Pin 12
2. Right Push Button - Digital Pin 9
3. Left LEDs - Digital Pin 4 and 5)
4. Right LEDs - Digital Pin 10 and 11



How it works

1. Push Button Sensing:
   - We have two push buttons connected to the Magicbit Tiny board (left button on pin 12 and right button on pin 9).
   - The buttons are configured as inputs with pull-up resistors (pull-up configuration)

Pull-up Configuration

With a pull-up resistor, the input pin will read a high state when the button is not pressed. In other words, a small amount of current is flowing between VCC and the input pin (not to ground), thus the input pin reads close to VCC. When the button is pressed, it connects the input pin directly to ground. The current flows through the resistor to ground, thus the input pin reads a low state.

2. LED Illumination:
   - Four LEDs are connected to different pins (left LEDs on pins 4, 5, and right LEDs on pins 10, 11).
   - Pressing the left button activates the left LEDs, and pressing the right button activates the right LEDs.

3. Control Logic:
   - When the left button is pressed, the left LEDs light up, and the right LEDs turn off.
   - When the right button is pressed, the right LEDs light up, and the left LEDs turn off.

Steps for the Program

STEP 01 Define the components pins

```
// Define component pins
const int leftButtonPin = 12;
const int rightButtonPin = 9;

const int leftLed1Pin = 4;
const int leftLed2Pin = 5;

const int rightLed1Pin = 10;
const int rightLed2Pin = 11;
```

STEP 02 Set up the pin modes

```
void setup() {
  // Configure push button pins as inputs with pull-up resistors
  pinMode(leftButtonPin, INPUT_PULLUP);
  pinMode(rightButtonPin, INPUT_PULLUP);

  // Configure LED pins as outputs
  pinMode(leftLed1Pin, OUTPUT);
  pinMode(leftLed2Pin, OUTPUT);
  pinMode(rightLed1Pin, OUTPUT);
  pinMode(rightLed2Pin, OUTPUT);
}
```

STEP 03 Define the functions for LEFT and RIGHT LEDs blinking

```
void lightUpLeftLEDs() {
  digitalWrite(leftLed1Pin, HIGH);  // Turn on left LED 1
  digitalWrite(leftLed2Pin, HIGH);  // Turn on left LED 2
  digitalWrite(rightLed1Pin, LOW);  // Turn off right LEDs
  digitalWrite(rightLed2Pin, LOW);
}

void lightUpRightLEDs() {
  digitalWrite(leftLed1Pin, LOW);  // Turn off left LEDs
  digitalWrite(leftLed2Pin, LOW);
  digitalWrite(rightLed1Pin, HIGH);  // Turn on right LED 1
  digitalWrite(rightLed2Pin, HIGH);  // Turn on right LED 2
}
```

STEP 04 Call the functions in a loop

```
void loop() {
  // Check if the left push button is pressed
  if (digitalRead(leftButtonPin) == LOW) {
    lightUpLeftLEDs();  // Light up left LEDs
  }

  // Check if the right push button is pressed
  if (digitalRead(rightButtonPin) == LOW) {
    lightUpRightLEDs();  // Light up right LEDs
  }
}
```

Full Program

Download the program code from here

STEP 05 Follow the steps below to upload example code for project anti-theft.

1. Connect Magicbit Tiny using the USB cable to a computer.
2. Open the Arduino IDE and get a New Sketch from the **File** menu.
3. Then in the **Tool** tab, select **Board -> Magicbit Tiny -> MagicBit Tiny**
4. In the same **Tool** tab select **port -> COM<your port number>**
5. Type or copy and paste above Arduino code to your new sketch and upload the code.

Learning Points:

- Press the left button to see the left LEDs light up or press the right button to witness the right LEDs illuminate. When the left button is pressed, the left LEDs light up, and the right LEDs turn off. When the right button is pressed, the right LEDs light up, and the left LEDs turn off.

Learning Points:

- **Push Button Sensing:** Understanding how push buttons can be used as inputs.
- **LED Control:** Learning how to control LEDs based on button input

Try to switch between different LED patterns using the push buttons.

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/xa3JW2JLj2E

07. Door Sensor

Create a program to make a DIY door sensor using Magicbit Tiny.

✦ Expected Output

- https://youtu.be/GzuNpUskyDY

✦ Description

- Welcome to the magical world of the Magicbit Tiny Door Sensor project! Let's explore the wonders of creating a door sensor using the Magicbit Tiny board. This project introduces the concept of infrared (IR) sensors and a buzzer, allowing you to build a simple and fun door sensor. It enables us to create a sensor that responds when a door is opened or closed. By using two infrared sensors and a buzzer, we can detect the door's movement and trigger a sound effect.

✦ Use components in Magicbit Tiny:

1. IR Sensor 1 - Pin A5 (Analog pin 5)
2. IR Sensor 2 - Pin A6 (Analog pin 6)
3. Buzzer - Digital Pin 13

How it works

1. IR Sensor Sensing:
   - Two IR sensors are connected to the Magicbit Tiny board (IR sensor 1 on pin A5 and IR sensor 2 on pin A6).
   - The sensors read analog values based on the presence of an object (e.g., a door) in their vicinity.

2. Buzzer Activation:
   - If the IR values from both sensors are less than 400, it indicates that the door is present (closed).
   - The buzzer is triggered to produce a tone, creating a doorbell-like effect.
   - After a short delay, the buzzer turns off.

Steps for the Program

STEP 01 Define the components pins

```
// Define component pins
const int irSensorPin1 = A5;
const int irSensorPin2 = A6;

const int buzzerPin = 13;
```

STEP 02 Configure the pins

```
void setup() {
  // Configure buzzer pin as output
  pinMode(buzzerPin, OUTPUT);
}
```

STEP 03 Make the process as a loop

```
void loop() {
  // Read the analog values from the IR sensors
  int irValue1 = analogRead(irSensorPin1);
  int irValue2 = analogRead(irSensorPin2);

  // Check if the IR value is greater than 400
  if (irValue1 > 900 || irValue2 > 900) {
    // Trigger the buzzer tone
    tone(buzzerPin, 2000);
    delay(200);
    noTone(buzzerPin);
  }
  delay(200);
}
```

Full Program

Download the program code from here

Follow the steps below to upload example code for project anti-theft.

1. Connect Magicbit Tiny using the USB cable to a computer.
2. Open the Arduino IDE and get a New Sketch from the **File** menu.
3. Then in the **Tool** tab, select **Board -> Magicbit Tiny -> MagicBit Tiny**
4. In the same **Tool** tab select **port -> COM<your port number>**
5. Type or copy and paste above Arduino code to your new sketch and upload the code.

Usage:

- Create a door / home structure using a piece of cardboard.
- Place the Magicbit Tiny which the above code uploaded near the door as shown in the figure below.
- Power the Magicbit Tiny using battery power.
- When the door is opened, the buzzer will produce a sound.
- Experiment by opening and closing the door to trigger the doorbell-like effect.

Learning Points:

- IR Sensor Sensing: Understanding how infrared sensors detect the presence of objects.
- Buzzer Activation: Learning how to use a buzzer to create sound effects based on sensor input.

Try adjusting the IR sensor sensitivity by modifying the threshold value. Experiment with different tones or melodies for the buzzer.

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/gm2MCABAlBg

08. DIY Sensor

ACTIVITY 08

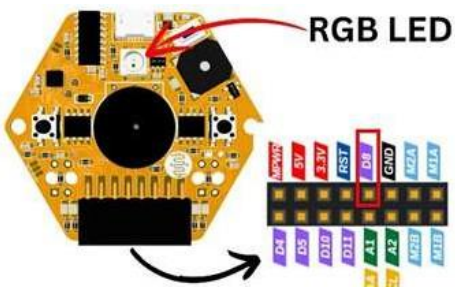
Create a program to indicate the conductivity of different materials as a Neo-pixel indicator using the Magicbit Tiny.

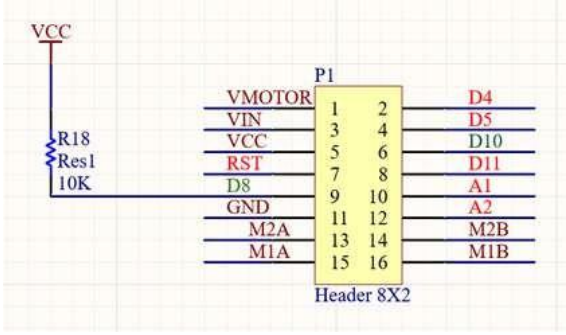
Description

- The project features a visual indicator using a single Neo-Pixel RGB LED, providing real-time feedback based on the conductivity value. The Neo-Pixel LED changes its color to represent different conductivity levels as well as moisture levels
- In this project we use Magicbit tiny in built pull up resistor.
- A pull-up resistor is used in electronic circuits to ensure a stable voltage level, typically high (1), when no other active device is driving the line low (0). It connects a signal line to the positive voltage supply, preventing it from floating or picking up stray signals. Pull-up resistors are essential for reliable input readings in microcontroller-based systems, ensuring defined logic levels. In tiny, the pull up resistor has been connected to pin 8.

Use components in Magicbit Tiny:

1. Tiny in built pull up resistor - Digital pin 8 (D8)
2. RGB LED – pin 0
3. 2 M-M jumper wires

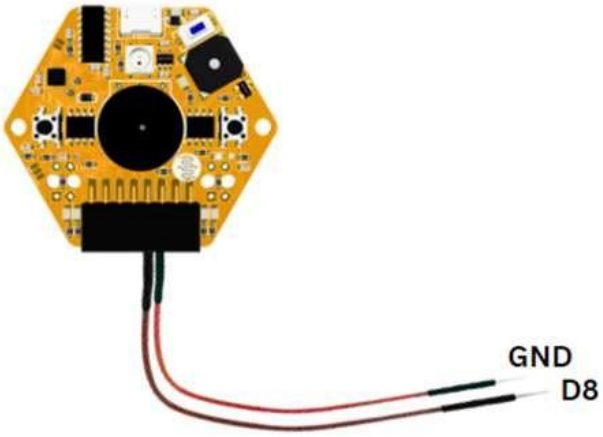




Special Note:

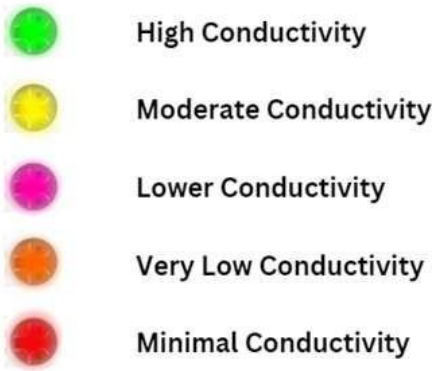
pin 8 can be act as a digital pin as well as an analog pin.

- Connect the two M-M jumper wires to the pin 8 and GND pins in the Magicbit Tiny as follows.



✦ How it works

1. Read Conductivity / moisture level:
   The analog value from pin 8 is read, representing the conductivity of the sensor circuit.
2. Convert to Voltage:
   The analog value is converted into a voltage to interpret the conductivity level accurately.
3. Color Coding:
   The Neo-Pixel RGB LED changes its color based on predefined conductivity and moisture ranges.



4. Display Update:

   The Neo-Pixel displays the relevant color.

✦ Steps for the Program

STEP 01 Include the required libraries and define the pins

```
#include <Adafruit_NeoPixel.h> // library to control the RGB LED

#define NUMPIXELS 1
#define PIN 0 // pin of the RGB LED

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

int condVal;
```

STEP 02 Make the setup

```
void setup() {
  Serial.begin(9600);
  pixels.begin();
}
```

STEP 03 Make the loop

```
void loop() {
  // Read analog value from pin 8
  condVal = analogRead(8);
  // Convert analog value to voltage
  float voltage = 5 - (condVal * (5.0 / 1023.0));
  // Set NeoPixel brightness
  pixels.setBrightness(255);
  // Color code based on conductivity value
  if (4 < voltage && voltage <= 5) {
    pixels.setPixelColor(0, pixels.Color(0, 255, 0));  // Green
    Serial.println("Conductivity value : " + String(voltage) + " Color : Green");
  } else if (3 < voltage && voltage <= 4) {
    pixels.setPixelColor(0, pixels.Color(255, 255, 0));  // Yellow
    Serial.println("Conductivity value : " + String(voltage) + " Color : Yellow");
  } else if (2 < voltage && voltage <= 3) {
    pixels.setPixelColor(0, pixels.Color(220, 160, 115));  // Pink
    Serial.println("Conductivity value : " + String(voltage) + " Color : Pink");
  } else if (1 < voltage && voltage <= 2) {
    pixels.setPixelColor(0, pixels.Color(255, 150, 0));  // Orange
    Serial.println("Conductivity value : " + String(voltage) + " Color : Orange");
  } else {
    pixels.setPixelColor(0, pixels.Color(255, 0, 0));  // Red
    Serial.println("Conductivity value : " + String(voltage) + " Color : Red");
  }
  // Update NeoPixel display
  pixels.show();
  // Delay for stability
  delay(50);
}
```

Full Program

[Download the program code from here]

Follow the steps below to upload example code for project anti-theft.

1. Connect Magicbit Tiny using the USB cable to a computer.
2. Open the Arduino IDE and get a New Sketch from the **File** menu.
3. Then in the **Tool** tab, select **Board -> Magicbit Tiny -> MagicBit Tiny**
4. In the same **Tool** tab select **port -> COM<your port number>**
5. Type or copy and paste the above Arduino code to your new sketch and upload the code.
6. Click on the icon on the top right corner of the Arduino IDE to open the Serial Monitor to view the conductivity value and corresponding color.

Observe the color changes on the Neo-Pixel LED based on the conductivity level.

Neo pixel RGB color for different conductivity level as follows.

Conductivity Level	Color
4-5	Green
3-4	Yellow
2-3	Pink
1-2	Orange
0-1	Red

Customization:

- Adjust the threshold voltage values in the code to customize conductivity levels and colors.
- Experiment with different Neo-Pixel patterns or multiple LEDs for enhanced visual feedback.

Learning Points:

- Understanding the relationship between conductivity and color.
- Hands-on experience with a simple conductivity sensor circuit.
- Introduction to interpreting analog sensor values.

09. Rainbow Light Magic

ACTIVITY 09

Create a program to witness a magical rainbow light pattern using the Neo-Pixel LED embedded in the Magicbit Tiny board.

✦ Expected Output

- https://youtu.be/akGnjOL4qwg

✦ Description

- The project features how the Magicbit Tiny board can create a mesmerizing rainbow light effect using its built-in Neo-Pixel LED. Let's dive into the simple documentation and uncover the secrets of this colorful project.

✦ Components

1. Magicbit Tiny Board
2. Built-in Neo-Pixel LED

✦ How it works:

1. **Neo-Pixel LED:**

   The Magicbit Tiny board features a built-in Neo-Pixel LED that can display a spectrum of colors.

2. **Color Pattern:**

   The project displays a captivating pattern of colors sequentially, creating a magical rainbow effect.

Steps for the Program

STEP 01 Include the required libraries and define the pins

```
#include <Adafruit_NeoPixel.h>
#define PIN 0           // Define the pin to which the NeoPixel is connected
#define NUMPIXELS 1      // Define the number of NeoPixels in your strip
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
```

STEP 02 Make the setup

```
void setup() {
  pixels.begin();  // Initialize the NeoPixel strip
}
```

STEP 03 Make the loop

```
void loop() {
  colorPattern(5);  // Change the number to adjust the speed of the pattern
}
// Display a pattern of colors sequentially
void colorPattern(uint8_t wait) {
  for (uint16_t color = 0; color < 256; color++) {
    pixels.setPixelColor(0, Wheel(color));
    pixels.show();
    delay(wait);
  }
}
// Input a value 0 to 255 to get a color value. The colors are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if (WheelPos < 85) {
    // Red to Green transition
    return pixels.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  }
  if (WheelPos < 170) {
    // Green to Blue transition
    WheelPos -= 85;
    return pixels.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
  // Blue to Red transition
  WheelPos -= 170;
  return pixels.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
}
```

Full Program

Download the program code from here

Follow the steps below to upload example code for project anti-theft.

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, Go to **tool** tab and select **Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. **Goto file -> New Sketch** and copy and paste the above Arduino code to your new sketch and upload the code.

Observe the mesmerizing rainbow light pattern as it unfolds automatically.

Learning Points:

- Neo-Pixel LED: Understanding the built-in programmable LED on the Magicbit Tiny board.
- Color Patterns: Exploring the creation of dynamic color patterns.

Experiment:

- Adjust the speed of the color pattern by changing the parameter in the `**colorPattern**` function.
- Now, let the Magicbit Tiny Rainbow Light Magic project fill your surroundings with vibrant colors and magic!

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/wdR_UOQrHts

10. People Count

ACTIVITY 10

Create a program to count the people using the Proximity IR sensors embedded in the Magicbit Tiny board.

Expected Output

- https://youtu.be/c93mD88_Z3g

Description

- The People Counting project is a simple yet effective application of infrared (IR) sensors with the Magicbit Tiny Board. Using two IR sensors, this project accurately counts the number of people entering and exiting a defined area. The system increments the count when someone enters and decrements the count when someone exits, providing real-time people counting capabilities.

Components

1. Magicbit Tiny Board
2. Infrared (IR) Sensor 1 (A5)
3. Infrared (IR) Sensor 2 (A6)

How it works:

1. **IR Sensor Sensing:**
   - IR Sensor 1 (A5) detects individuals entering.
   - IR Sensor 2 (A6) detects individuals exiting.

2. IR Sensor Sensing:

- When IR Sensor 1 reads a value below 700, the people count increments, indicating an entry.
- When IR Sensor 2 reads a value below 700, the people count decrements, indicating an exit.

✦ Steps for the Program

STEP 01 Define the components pins

```
const int irSensorPin1 = A5;
const int irSensorPin2 = A6;
int peopleCount = 0;
```

STEP 02 Make the setup / serial monitor

```
void setup() {
  Serial.begin(9600);
}
```

STEP 03 Arrange the loop

```
void loop() {
  int irValue1 = analogRead(irSensorPin1);
  int irValue2 = analogRead(irSensorPin2);
 // Check if A5 reads a value less than 700, increment people count
  if (irValue1 < 700) {
    peopleCount++;
    Serial.println("Person Entered!");
    Serial.print("People Count: ");
    Serial.println(peopleCount);
    delay(1000); // Time taken to pass both sensors
  }
// Check if A6 reads a value less than 700, decrement people count
  if (irValue2 < 700) {
    peopleCount--;
// Ensure people count doesn't go below zero
    if (peopleCount < 0) {
      peopleCount = 0;
    } else {
      Serial.println("Person Exited!");
      Serial.print("People Count: ");
      Serial.println(peopleCount);
    }
    delay(1000); // Time taken to pass both sensors
  }
  delay(100);  // Delay for stability
}
```

Full Program

Download the program code from here

Follow below steps to upload example code for project anti-theft.

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, Go to **tool** tab and **select Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. **Goto file -> New Sketch** and copy and paste above Arduino code to your new sketch and upload the code.

Place the Magicbit Tiny People Count system at an entrance and witness the count increase with entries and decrease with exits.

Learning Points:

- IR Sensor Operation:
  Understand how IR sensors detect individuals passing through.
- Logical Counting:
  Learn the basics of counting logic based on sensor readings.

Adjust the threshold value (700) and observe how it affects the sensitivity of the people count system.

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/bFmAKzFzATA

11. Motor Controlling

ACTIVITY 11

Create a program to control the speed and direction of two motors using a potentiometer.

✦ Expected Output

- https://youtu.be/22XTVQAdaRY

✦ Description

- This project introduces you to the basics of motor control using the built-in motor driver L9110S on the Magicbit Tiny board. Let's dive into the simple documentation and explore the wonders of this engaging project.

✦ Components

1. Magicbit Tiny Board
2. Built-in Motor Driver L9110S (2x)
3. Built-in Potentiometer
4. 3.7 V battery
5. DC Motors (2x)

✦ How it works:

1. **Potentiometer Input:**
   - The potentiometer (connected to analog pin A3) reads user input.
   - Potentiometer values are mapped to a range of motor speeds (-255 to 255).
2. **Motor Control:**
   - The built-in motor driver L9110S simplifies motor control.
   - Motor 1 and Motor 2 speed and direction are adjusted based on the potentiometer input.

3. **Potentiometer Input:**
   - The project provides optional serial monitor output, displaying potentiometer values and motor speeds for further exploration.

✦ Steps for the Program

STEP 01 Define the pins and variables

```
// Motor 1
const int motor1Input1 = 4;  // Input 1 pin for Motor 1
const int motor1Input2 = 5;  // Input 2 pin for Motor 1
// Motor 2
const int motor2Input1 = 10; // Input 1 pin for Motor 2
const int motor2Input2 = 11; // Input 2 pin for Motor 2
// Potentiometer
const int potPin = A3;       // Potentiometer connected to analog pin A3
```

STEP 02 Make the setup

```
void setup() {
  Serial.begin(9600);        // Start serial communication for debugging
}
```

STEP 03 Make the loop

```
void loop() {
  // Read the potentiometer value
    int potValue = analogRead(potPin);
  // Map the potentiometer value to a range of motor speeds (-255 to 255)
    int motorSpeed = map(potValue, 0, 1023, -255, 255);
 // Control Motor 1 (Left Motor)
  if (motorSpeed >= 0) {
    analogWrite(motor1Input1, motorSpeed);
    analogWrite(motor1Input2, 0);
    analogWrite(motor2Input1, motorSpeed);
    analogWrite(motor2Input2, 0);
  } else {
    analogWrite(motor1Input1, 0);
    analogWrite(motor1Input2, -motorSpeed);
    analogWrite(motor2Input1, 0);
    analogWrite(motor2Input2, -motorSpeed);
  }
  //Print potentiometer value and motor speed to the serial monitor (optional)
  Serial.print("Potentiometer: ");
  Serial.print(potValue);
  Serial.print("   Motor Speed: ");
  Serial.println(motorSpeed);
  // Add a delay to avoid rapid changes (adjust as needed)
  delay(100);
}
```

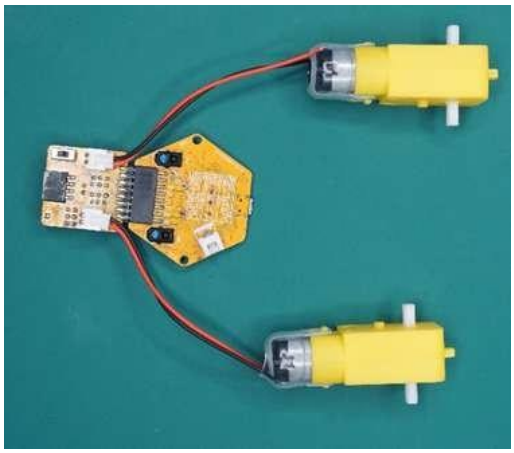
Full Program

Download the program code from here

Follow the steps below to upload example code for tiny motor controlling project

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, Go to **tool** tab and select **Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. Go to **file -> New Sketch** and copy and paste above Arduino code to your new sketch and upload the code.

- After uploading the code, remove the USB connector.
- Connect the extension module to the Magicbit Tiny.
- Connect a 3.7V battery to the battery connector on the Extension.
- Connect the motors to the motor connectors in the extension and turn on the switch.
- Experiment with different potentiometer values and observe how motor behavior changes. Try adjusting the delay duration for a different responsiveness.



Learning Points:

- Introduction to Motor Drivers: Understanding the role of the built-in L9110S motor driver.
- Potentiometer Interaction: Learning how to use a potentiometer for input.
- Motor Speed Control: Exploring how to control motor speed based on user input.

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/fU8y88mVCzw

12. Obstacle Avoiding Car

To build a robot car that can navigate while avoiding obstacles.

✦ Expected Output

- https://youtu.be/LPN8Rmhk8iM

✦ Description

- This project introduces you to the exciting combination of an ultrasonic sensor and motor control on the Magicbit Tiny board. It enables you to build a robot that can navigate and avoid obstacles using an ultrasonic sensor. This project leverages motor control to enable the robot to move forward and turn right based on the sensor readings.

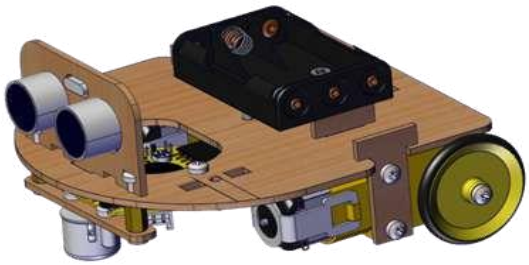
✦ How it works:

1. **Ultrasonic Sensor:**
   - The HC-SR04 ultrasonic sensor is used to measure the distance to obstacles.
2. **Motor Control:**
   - DC motors are controlled using the built-in motor driver L9110S.
   - Motor functions include moving forward, backward, stopping, and turning right.
3. **Obstacle Avoidance:**
   - If an obstacle is detected within a certain range (e.g., 10 cm), the robot turns right to avoid it.
4. **Serial Monitor (Optional):**
   - Distance readings are displayed on the Serial Monitor for monitoring and debugging.

✦ Set-Up:

- Build the robot car structure according to the steps given in the below video.

Robot Assembling Guide - https://youtu.be/q6y0NJX4XmA



✦ Steps for the Program

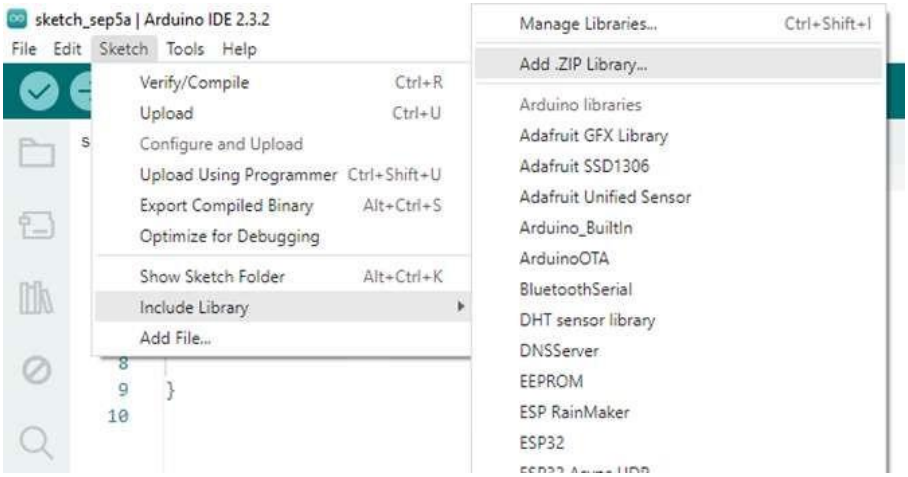
STEP 01 Define the libraries

```
#include <Ultrasonic.h>
```

This library (used for ultrasonic sensor with Magicbit Tiny) is a custom made library .

[Download the Ultrasonic Library Here]

Install the **"Ultrasonic.h"** library to your Arduino interface before you run the below program.



STEP 02 Define the pins and variables

```
Ultrasonic ultrasonic(A1);
const int motor1A = 5;   // Motor 1, input A (Right)
const int motor1B = 4;   // Motor 1, input B
const int motor2A = 10;  // Motor 2, input A (Left)
const int motor2B = 11;  // Motor 2, input B

int distance;
```

STEP 03 Make the setup

```
void setup() {
  // Motor control pins as outputs
  pinMode( motor1A, OUTPUT);
  pinMode( motor1B, OUTPUT);
  pinMode( motor2A, OUTPUT);
  pinMode( motor2B, OUTPUT);

  Serial.begin(9600);
  }
```

STEP 04 Make the loop

```
void loop(){

  distance = ultrasonic.read();
  // Print the distance to the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  // Check if there is an obstacle within a certain range
  if (distance < 10 && distance > 0) {
  // If obstacle detected, turn right
    turnRight();
  } else {
  // If no obstacle, move forward
    moveForward();
    }
  delay(10);  // Add a small delay for stability
}
```

STEP 05 Make the functions for forward and right or left motions

```
// Function to move the robot forward
void moveForward() {
  digitalWrite(motor1A, LOW);
  digitalWrite(motor1B, HIGH);
  digitalWrite(motor2A, LOW);
  digitalWrite(motor2B, HIGH);
}
  // Function to turn the robot right
void turnRight() {
  digitalWrite(motor1A, HIGH);
  digitalWrite(motor1B, LOW);
  digitalWrite(motor2A, LOW);
  digitalWrite(motor2B, HIGH);
}
```

Full Program

Download the program code from here

Follow below steps to upload example code for obstacle avoiding robot car project.

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, Go to **tool** tab and select **Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. **Goto file -> New Sketch** and copy and paste above Arduino code to your new sketch and upload the code.

Learning Points:

- Ultrasonic Sensor Usage: Understanding how the HC-SR04 measures distance.
- Motor Control Logic: Learning how to control DC motors for various movements.
- Obstacle Avoidance: Exploring the concept of using sensors to avoid obstacles.

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/1mzosiE-jWc

13. Line Follower Car

ACTIVITY 13

To build a robot car that can navigate while following a line / track.

✦ Expected Output

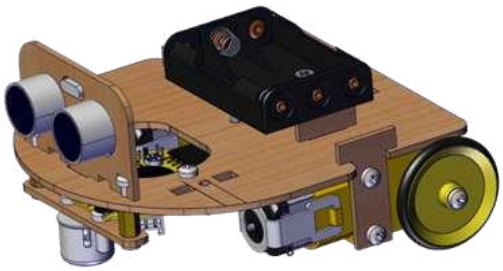
- https://youtu.be/nZds7kcdqco

✦ Description

- The Magicbit Tiny Line Following Robot project allows you to create a robot that can follow a line using infrared sensors. By adjusting motor speeds based on sensor readings, the robot can navigate along a predefined path.

✦ How it works:

1. **Infrared Sensors:**
   - Two built-in IR sensors are used to detect the line beneath the robot.
2. **Motor Control:**
   - DC motors are controlled using the built-in motor driver L9110S.
   - Motor speeds are adjusted based on the readings from the IR sensors.
3. **Line Following Logic:**
   - The robot adjusts its movement to stay on the line, using the difference in sensor reading.
4. **Serial Monitor (Optional):**
   - Sensor values and motor speeds are displayed on the Serial Monitor for monitoring and debugging.



✦ Set-Up:

- Build the robot car structure according to the steps given in the below video. (It is the same robot car you used in Obstacle Avoiding Car Activity)

  Robot Assembling Guide - https://youtu.be/q6y0NJX4XmA

✦ Steps for the Program

STEP 01 Define the component pins

```
// Define motor pins
  int motor1Pin1 = 11;  // Motor 1 input pin 1
  int motor1Pin2 = 10;  // Motor 1 input pin 2
  int motor2Pin1 = 5;   // Motor 2 input pin 1
  int motor2Pin2 = 4;   // Motor 2 input pin 2

  // Define sensor pins
  int rightSensorPin = A5;  // Right sensor pin
  int leftSensorPin = A6;   // Left sensor pin
```

STEP 02 Make the setup

```
void setup() {
// Set motor pins as outputs
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(motor2Pin1, OUTPUT);
  pinMode(motor2Pin2, OUTPUT);
// Initialize serial communication for monitoring
  Serial.begin(9600);
}
```

STEP 03 Make the loop

```
void loop() {
// Read sensor values
  int rightSensorValue = analogRead(rightSensorPin);
  int leftSensorValue = analogRead(leftSensorPin);
// Print sensor values
  Serial.print("Right Sensor: ");
  Serial.print(rightSensorValue);
  Serial.print("\t");
  Serial.print("Left Sensor: ");
  Serial.print(leftSensorValue);
  Serial.print("\t");
// Adjust motor speeds based on sensor values
  int speedDifference = rightSensorValue - leftSensorValue;
// Set motor speeds
  int baseSpeed = 200;  // You can adjust this value based on your robot's speed requirements
// Adjust motor speeds based on sensor readings
  int motor1Speed = baseSpeed + speedDifference;
  int motor2Speed = baseSpeed - speedDifference;
// Make sure motor speeds are within valid range (0 to 255)
  motor1Speed = constrain(motor1Speed, 0, 195);
  motor2Speed = constrain(motor2Speed, 0, 195);
// Control the motors
  analogWrite(motor1Pin1, motor1Speed);
  analogWrite(motor1Pin2, 0);  // Set the other direction pin to 0
  analogWrite(motor2Pin1, 0);
  analogWrite(motor2Pin2, motor2Speed );
// Print motor speeds for monitoring
  Serial.print("motor1: ");
  Serial.print(motor1Speed);
  Serial.print("\t");
  Serial.print("motor2: ");
  Serial.println(motor2Speed);
// Add a delay to control the loop speed
  delay(10);
}
```

Full Program

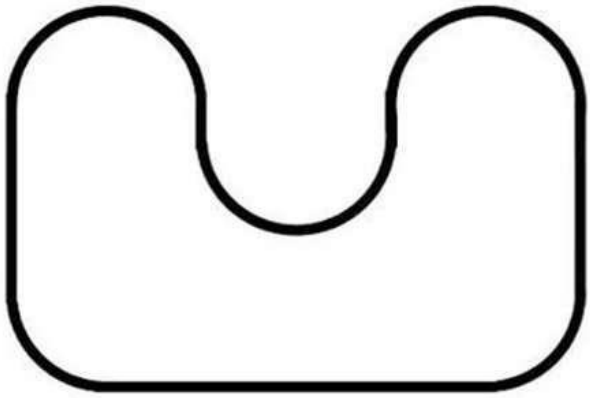
[Download the program code from here]

Follow below steps to upload example code for obstacle avoiding robot car project.

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, Go to **tool** tab and select **Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. Go to **file -> New Sketch** and copy and paste above Arduino code to your new sketch and upload the code.

Prepare a line following track (Black line in a white surface or White line in a Black surface / Line width [2cm -3 cm]).

[Download printable Line Follower Track]



Important - This document is scaled to A2 size.

Learning Points:

- IR Sensor Usage: Understanding how IR sensors detect variations in reflectivity.
- Motor Control for Line Following: Learning how to control DC motors to navigate along a line.
- Logic for Line Following: Exploring the concept of adjusting motor speeds based on sensor readings.

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/2zW-tE3mnxI

14. Robot Roach

ACTIVITY 14

To build a robot car that is sensitive to light intensity.

✦ Expected Output

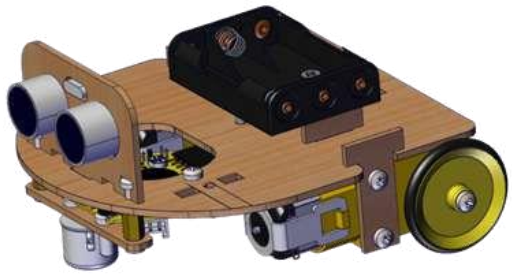
- https://youtu.be/hP4Kq4Hnfe0

✦ Description

- This project introduces you to the exciting combination of an LDR sensor and motor control on the Magicbit Tiny board. It enables you to build a robot that mimics the behavior of a cockroach by moving away from light sources detected by the LDR. This project leverages motor control to enable the robot to steer and move accordingly based on the light intensity readings.

✦ How it works:

1. **Infrared Sensors:**
   - Two built-in IR sensors are used to detect the line beneath the robot.
2. **Motor Control:**
   - DC motors are controlled using the built-in motor driver L9110S.
   - Motor speeds are adjusted based on the readings from the IR sensors.
3. **Line Following Logic:**
   - The robot adjusts its movement to stay on the line, using the difference in sensor reading.
4. **Serial Monitor (Optional):**
   - Sensor values and motor speeds are displayed on the Serial Monitor for monitoring and debugging.



✦ Set-Up:

- Build the robot car structure according to the steps given in the below video. (It is the same robot car you used in Obstacle Avoiding Car Activity)

  Robot Assembling Guide - https://youtu.be/q6y0NJX4XmA

✦ Steps for the Program

STEP 01 Define the component pins

```
// Define motor pins
const int motor1A = 5;   // Motor 1, input A (Right)
const int motor1B = 4;   // Motor 1, input B
const int motor2A = 10;  // Motor 2, input A (Left)
const int motor2B = 11;  // Motor 2, input B

// Define LDR sensor pin
#define LDR_PIN A7

// Define light threshold to detect light
#define LIGHT_THRESHOLD 600  // Adjust this threshold as needed in between 0 - 1023
```

STEP 02 Make the setup

```
void setup() {
  // Motor control pins as outputs
  pinMode( motor1A, OUTPUT);
  pinMode( motor1B, OUTPUT);
  pinMode( motor2A, OUTPUT);
  pinMode( motor2B, OUTPUT);

  // LDR as input
  pinMode(LDR_PIN,INPUT);

  Serial.begin(9600);
}
```

STEP 03 Make the loop

```
void loop() {
  // Read the LDR sensor value
  int lightLevel = analogRead(LDR_PIN);

  // Print the LDR sensor value to the Serial Monitor for debugging
  Serial.print("Light Level: ");
  Serial.println(lightLevel);

  // Check if light is detected
  if (lightLevel > LIGHT_THRESHOLD) {
    // Light detected, move away from light (reverse or turn)
    //moveBackward();
    //delay(500); // Move backward for 0.5 seconds
    turnLeft();
    delay(500);
    moveForward();
     // Turn left for 0.5 seconds
  } else {
    // No light detected, move forward
    moveForward();
  }
}
```

```
// Function to move the robot forward
void moveForward() {
  digitalWrite(motor1A, LOW);  // Right motor forward
  digitalWrite(motor1B, HIGH);   // Right motor backward
  digitalWrite(motor2A, LOW);  // Left motor forward
  digitalWrite(motor2B, HIGH);   // Left motor backward
}

// Function to move the robot backward
void moveBackward() {
  digitalWrite(motor1A, HIGH);   // Right motor forward
  digitalWrite(motor1B, LOW);  // Right motor backward
  digitalWrite(motor2A, HIGH);   // Left motor forward
  digitalWrite(motor2B, LOW);  // Left motor backward
}
```

```
// Function to turn the robot left
void turnLeft() {
  digitalWrite(motor1A, LOW);   // Right motor forward
  digitalWrite(motor1B, HIGH);  // Right motor backward
  digitalWrite(motor2A, HIGH);  // Left motor forward
  digitalWrite(motor2B, LOW);   // Left motor backward
}

// Function to turn the robot right
void turnRight() {
  digitalWrite(motor1A, HIGH);  // Right motor forward
  digitalWrite(motor1B, LOW);   // Right motor backward
  digitalWrite(motor2A, LOW);   // Left motor forward
  digitalWrite(motor2B, HIGH);  // Left motor backward
}
```

Full Program

Download the program code from here

Follow the steps below to upload example code for Roach Robot project.

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, Go to **tool** tab and select **Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. Go to **file -> New Sketch** and copy and paste above Arduino code to your new sketch and upload the code.

    - Open the Serial Monitor in Arduino IDE and see how the light level readings vary according to the light level in your background.
    - Adjust the threshold value in the code according to that.
    - Upload the final code and remove the USB cable while powering via batteries.

Direct a torch light or flashlight to increase the light intensity and observe how robot avoid the lights.

15. Dancing Robot

ACTIVITY 15

To build a robot car that behaves like dancing to music.

✦ Expected Output

- https://youtu.be/1p1UN6Dc_14

✦ Description

- This project introduces you to the exciting combination of motor control, a buzzer, and Neo-pixel LEDs on the Magicbit Tiny board. It enables you to build a robot that mimics a dancing robot by performing various movements, generating musical tones through the buzzer, and displaying vibrant color patterns using the Neo-pixel LEDs. This project leverages motor control to choreograph the robot's movements, synchronized with dynamic audio and visual effects to create an engaging and interactive experience.

✦ How it works:

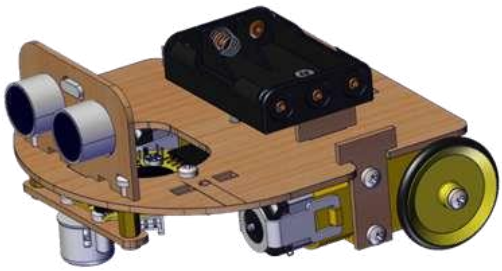
1. **Neo-pixel LEDs**
   - The Neo-pixel LEDs are used to create dynamic lighting effects, displaying a variety of colors and patterns to enhance the dancing experience.
   - The colors and patterns are synchronized with the robot's movements and the musical tones.
2. **Motor Control:**
   - DC motors are controlled using the built-in motor driver on the Magicbit Tiny board.
   - Motor functions include performing different movements like spinning, swaying, or stepping in sync with the music and lighting effects.

1. **Neo-pixel LEDs**
   - The Neo-pixel LEDs are used to create dynamic lighting effects, displaying a variety of colors and patterns to enhance the dancing experience.
   - The colors and patterns are synchronized with the robot's movements and the musical tones.
2. **Motor Control:**
   - DC motors are controlled using the built-in motor driver on the Magicbit Tiny board.
   - Motor functions include performing different movements like spinning, swaying, or stepping in sync with the music and lighting effects.
3. **Musical Tones:**
   - The onboard buzzer generates musical tones to accompany the robot's movements.
   - The tones can be customized to create rhythmic or melodic sequences for a more engaging performance.

✦ Set-Up:

- Build the robot car structure according to the steps given in the below video. (It is the same robot car you used in Obstacle Avoiding Car Activity)

Robot Assembling Guide - https://youtu.be/q6y0NJX4XmA



Steps for the Program

STEP 01 Add the required Libraries

```
#include <Adafruit_NeoPixel.h>
```

STEP 02 Define the component pins

```
// Define pin assignments
#define MOTOR1_IN1 4
#define MOTOR1_IN2 5
#define MOTOR2_IN1 10
#define MOTOR2_IN2 11
#define NEOPIXEL_PIN 0
#define BUZZER_PIN 13

// NeoPixel setup
#define NUMPIXELS 1
Adafruit_NeoPixel pixels(NUMPIXELS, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);

// Function prototypes
void moveLeft();
void moveRight();
void moveForward();
void moveBackward();
void stopMotors();
void setNeoPixelColor(int red, int green, int blue);
void playTone(int frequency, int duration);
```

STEP 03 Make the setup

```
void setup() {
  // Initialize motor pins
  pinMode(MOTOR1_IN1, OUTPUT);
  pinMode(MOTOR1_IN2, OUTPUT);
  pinMode(MOTOR2_IN1, OUTPUT);
  pinMode(MOTOR2_IN2, OUTPUT);

  // Initialize NeoPixel
  pixels.begin();

  // Initialize buzzer
  pinMode(BUZZER_PIN, OUTPUT);
}
```

STEP 04 Create the other functions

```
// Function to move the car left
void moveLeft() {
  // Left wheel moves backward, right wheel moves forward
  analogWrite(MOTOR1_IN1, 0);
  analogWrite(MOTOR1_IN2, 255);
  analogWrite(MOTOR2_IN1, 255);
  analogWrite(MOTOR2_IN2, 0);

  // Set NeoPixel colors and play tones
  setNeoPixelColor(255, 0, 0); // Red
  playTone(262, 500); // Play C4 note
  delay(500);

  setNeoPixelColor(0, 255, 0); // Green
  playTone(294, 500); // Play D4 note
  delay(500);

  setNeoPixelColor(0, 0, 255); // Blue
  playTone(330, 500); // Play E4 note
  delay(500);
}
```

```
// Function to move the car right
void moveRight() {
  // Left wheel moves forward, right wheel moves backward
  analogWrite(MOTOR1_IN1, 255);
  analogWrite(MOTOR1_IN2, 0);
  analogWrite(MOTOR2_IN1, 0);
  analogWrite(MOTOR2_IN2, 255);

  // Set NeoPixel colors and play tones
  setNeoPixelColor(255, 255, 0); // Yellow
  playTone(349, 500); // Play F4 note
  delay(500);

  setNeoPixelColor(255, 0, 255); // Magenta
  playTone(392, 500); // Play G4 note
  delay(500);

  setNeoPixelColor(0, 255, 255); // Cyan
  playTone(440, 500); // Play A4 note
  delay(500);
}
```

```
// Function to move the car forward
void moveForward() {
  // Both wheels move forward
  analogWrite(MOTOR1_IN1, 255);
  analogWrite(MOTOR1_IN2, 0);
  analogWrite(MOTOR2_IN1, 255);
  analogWrite(MOTOR2_IN2, 0);

  // Set NeoPixel colors and play tones
  setNeoPixelColor(255, 165, 0); // Orange
  playTone(494, 500); // Play B4 note
  delay(500);

  setNeoPixelColor(128, 0, 128); // Purple
  playTone(523, 500); // Play C5 note
  delay(500);

  setNeoPixelColor(255, 20, 147); // Deep Pink
  playTone(587, 500); // Play D5 note
  delay(500);
}
```

```
// Function to move the car backward
void moveBackward() {
  // Both wheels move backward
  analogWrite(MOTOR1_IN1, 0);
  analogWrite(MOTOR1_IN2, 255);
  analogWrite(MOTOR2_IN1, 0);
  analogWrite(MOTOR2_IN2, 255);

  // Set NeoPixel colors and play tones
  setNeoPixelColor(75, 0, 130); // Indigo
  playTone(659, 500); // Play E5 note
  delay(500);

  setNeoPixelColor(255, 69, 0); // Red-Orange
  playTone(698, 500); // Play F5 note
  delay(500);

  setNeoPixelColor(0, 128, 128); // Teal
  playTone(784, 500); // Play G5 note
  delay(500);
}
```

```
// Function to stop the motors
void stopMotors() {
  analogWrite(MOTOR1_IN1, 0);
  analogWrite(MOTOR1_IN2, 0);
  analogWrite(MOTOR2_IN1, 0);
  analogWrite(MOTOR2_IN2, 0);
}

// Function to set NeoPixel color
void setNeoPixelColor(int red, int green, int blue) {
  pixels.setPixelColor(0, pixels.Color(red, green, blue));
  pixels.show();
}

// Function to play tone
void playTone(int frequency, int duration) {
  tone(BUZZER_PIN, frequency, duration);
  delay(duration);
  noTone(BUZZER_PIN);
}
```

Full Program

Download the program code from here

Follow the steps below to upload example code for Roach Robot project.

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, Go to **tool** tab and select **Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. **Goto file -> New Sketch** and copy and paste the above Arduino code to your new sketch and upload the code.

In the above program, you can change the colors according to their relevant RGB values.

In the above program, you can change the playing tones by changing the relevant frequencies for different notes.

Refer the code in **Activity 03 – Melody Magic** for different nots and their respective frequencies.

16. Path Clearing Robot

To build a robot car that can navigate while clearing his path.

✦ Expected Output

- https://youtu.be/31B12WFps18

✦ Description

- The Magicbit Tiny Path Clearing Robot project enables you to build a robot that can navigate and detect obstacles in his path using an ultrasonic sensor. And then by operating the servo motor it can grab the detected object and remove it from his path and continue robot's motion. This project leverages motor control to enable the robot to move forward and turn Right / Left based on the sensor readings and operate the servo motor.

✦ How it works:

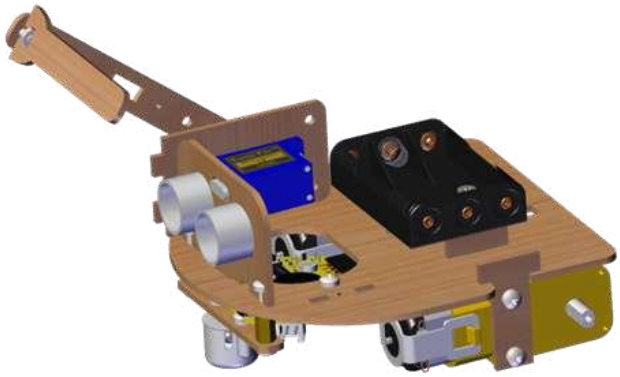
1. **Ultrasonic Sensor:**
   - The HC-SR04 ultrasonic sensor is used to measure the distance to obstacles.
2. **Motor Control:**
   - DC motors are controlled using the built-in motor driver L9110S.
   - Motor functions include moving forward, backward, stopping, and turning right / left.
3. **Grabbing the objects:**
   - If an obstacle is detected within a certain range (e.g., 10 cm), the robot operates its arm by operating the servo motor to grab the object and remove it from the moving path.

4. **Serial Monitor (Optional):**
   - The Distance readings are displayed on the Serial Monitor for monitoring and debugging.

✦ Set-Up:

- Use the same robot used for Obstacle avoiding and line following robot with below modifications.

Robot Assembling Guide - https://youtu.be/hq_ozjGHdt8



✦ Steps for the Program

STEP 01 Add the required Libraries

```
#include <Ultrasonic.h>
#include <TinyServo.h>
```

STEP 02 Define the component pins

```
const int motor1A = 5;   // Motor 1, input A
const int motor1B = 4;   // Motor 1, input B
const int motor2A = 10;  // Motor 2, input A
const int motor2B = 11;  // Motor 2, input B

Servo myservo;  // create servo object to control a servo
int pos = 60;    // variable to store the servo position

Ultrasonic ultrasonic(A1);
int distance;
```

STEP 03 Make the setup

```
void setup() {
  Serial.begin(9600);  // Initialize serial communication
  myservo.attach(8);  // attaches the servo on pin 8 to the servo object
  pinMode(motor1A, OUTPUT);
  pinMode(motor1B, OUTPUT);
  pinMode(motor2A, OUTPUT);
  pinMode(motor2B, OUTPUT);
}
```

STEP 04 Make the loop

```
void loop() {

  distance = ultrasonic.read();
  // Print the distance to the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  // Check if there is an obstacle within a certain range
  if (distance < 10 && distance > 0) {
   stopMotors();
   delay(500);
   moveForward();
   delay(300);
   stopMotors();
   delay(500);

  for (pos = 60; pos >= 0; pos -= 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }


  delay(700);
  moveBackward();
  delay(1200);
  stopMotors();
  delay(300);
  turnleft();
  delay(450);
  stopMotors();
  delay(300);

  for (pos = 0; pos <= 60; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
```

```
    stopMotors();
    delay(100);
    moveBackward();
    delay(250);
    stopMotors();
    delay(300);
    turnright();
    delay(480);
    stopMotors();
    delay(300);
    moveForward();
    delay(500);
    }
    else {
      // If no obstacle, move forward
      moveForward();
    }
    delay(10);  // Add a small delay for stability
}
```

STEP 05 Create the other functions

```
void moveBackward() {
  // Move forward
  digitalWrite(motor1A, HIGH);
  digitalWrite(motor1B, LOW);
  digitalWrite(motor2A, HIGH);
  digitalWrite(motor2B, LOW);
}

void moveForward() {
  // Move forward
  digitalWrite(motor1A, LOW);
  digitalWrite(motor1B, HIGH);
  digitalWrite(motor2A, LOW);
  digitalWrite(motor2B, HIGH);
}

void stopMotors() {
  // Stop motors
  digitalWrite(motor1A, LOW);
  digitalWrite(motor1B, LOW);
  digitalWrite(motor2A, LOW);
  digitalWrite(motor2B, LOW);
}

void turnright() {
  digitalWrite(motor1A, HIGH);
  digitalWrite(motor1B, LOW);
  digitalWrite(motor2A, LOW);
  digitalWrite(motor2B, HIGH);
}

void turnleft() {
  digitalWrite(motor1A, LOW);
  digitalWrite(motor1B, HIGH);
  digitalWrite(motor2A, HIGH);
  digitalWrite(motor2B, LOW);
}
```

Follow the steps below to upload example code for path clearing robot car project.

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, Go to **tool** tab and select **Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. Go to **file -> New Sketch** and copy and paste above Arduino code to your new sketch and upload the code.

Important - Install the **"Ultrasonic.h"** library to your Arduino interface before you run the below program.

The library used for ultrasonic sensor with Magicbit Tiny is a custom made library.

Download the Ultrasonic Library Here

Observe how the robot navigates its environment, avoiding obstacles based on the ultrasonic sensor readings.

No need of installing **"TinyServo.h"** library manually as it is already installed with Magicbit Tiny Board.

Full Program

Download the program code from here

17. Automatic Rail Gate

To create a program to operate the wooden arm connected servo motor as a rail gate according to the detection of the train.

Description

- This project introduces you to the exciting combination of Proximity IR sensor and servo motor control on the Magicbit Tiny board. It enables you to build a setup that mimics the behavior of a rail gate which block the road when a train is to cross the road and once the train passes the road, the gate get open back allowing the vehicles to cross the rail way.

Components

- Magicbit Tiny Board
- Tiny extension module
- USB cable
- M-M jumper wires
- Cardboard
- Servo motor
- Printed Track
- Wooden servo arm

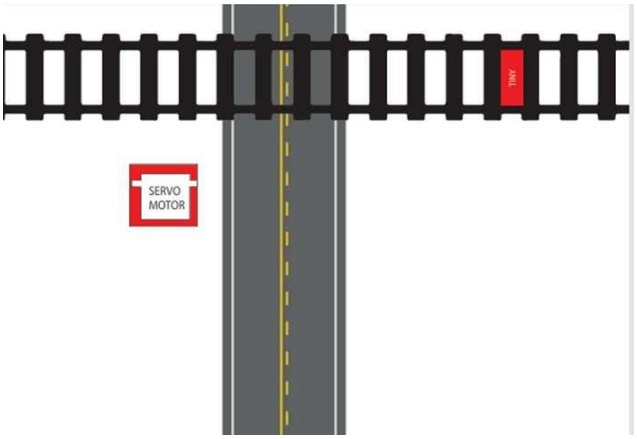
How it works:

1. **Proximity IR Sensor**
   - IR sensors detect the presence of an object (train).

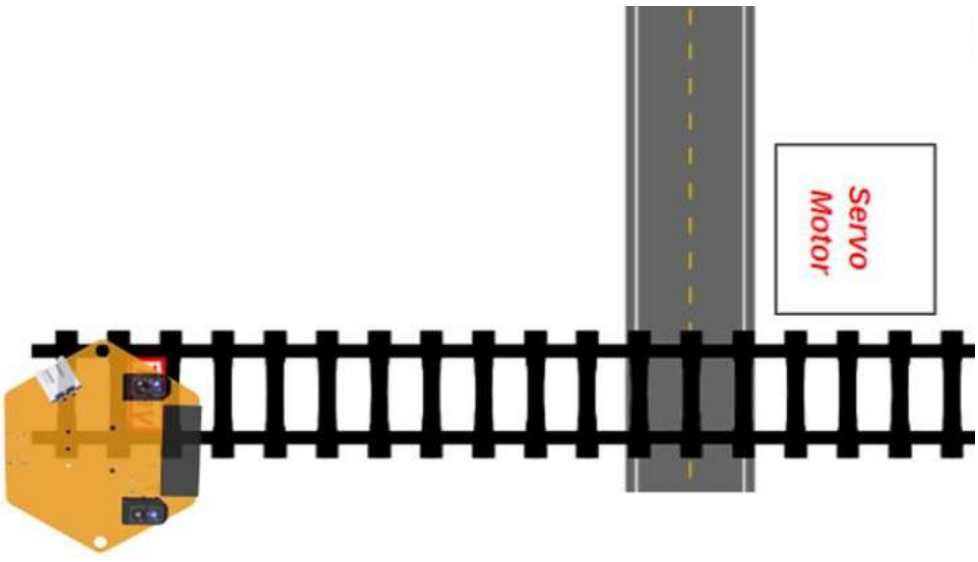
2. **Servo Motor:**
   - The servo motor regulates the gate's opening and closing mechanism.
   - Upon detecting the train, the servo motor rotates to the given position (angle) making the gate close.
   - After a given time period, it rotates back to another given position making the gate opened.
   - Time period is decided according to the time taken by the train to pass the gate

✦ Set-Up:

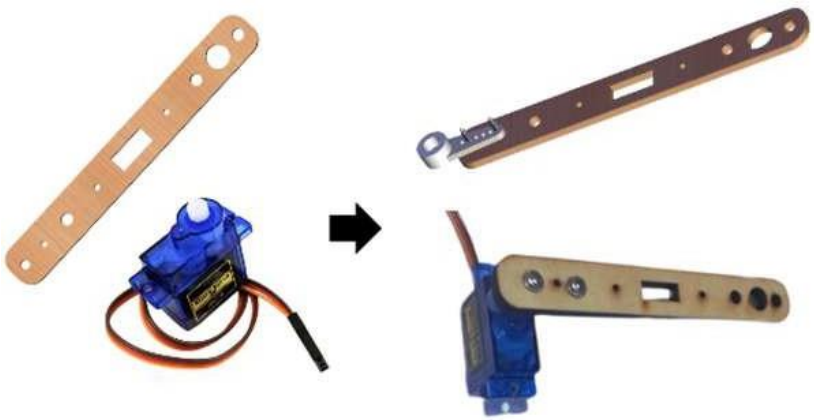
Take a printout of the below track - Print the Track



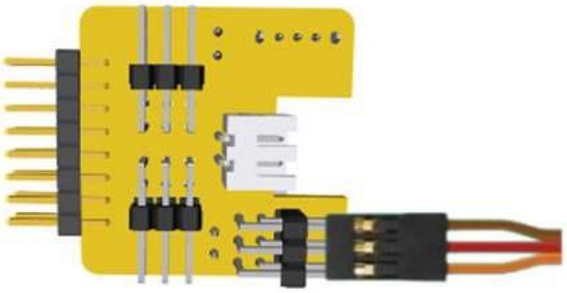
- Paste the printed track on piece of cardboard and remove the area marked as "Tiny"
- Place Magicbit tiny as below.



- Fix the wooden servo arm with the servo motor.



- Attach the servo motor in the marked area of the set up at a higher level .
- Connect the servo motor cable with tiny extension at pin 8



- Connect the extension with Magicbit Tiny and power the whole set up via batteries by connecting the battery connector to the extension. Turn ON the switch in the extension.
- Set-up creation guidelines - https://youtu.be/gxgTAXWFNac

Get the positions of the servo motor

Follow the steps below to upload example code for path clearing robot car project.

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, Go to **tool** tab and select **Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. Go to **file -> New Sketch** and copy and paste below Arduino code to your new sketch and upload the code.

Download the program code from here

In the above code, change the value "90" to any value in between 0 - 180 and identify the two values corresponding to the gate opened position and gate closed position.

Upload the below program to operate the servo motor according to the motion detected by the IR proximity sensor in the Magicbit Tiny.

Download the code

- Upload the above code and move your hand over the IR sensors and observe how the reading get vary due to detection of your hand.
- Depending on that, change the threshold value in the above code and re-upload the code.
- Observe how the system operates.

18. Fire Detector

ACTIVITY 18

To create a program to detect fire using proximity IR sensors and indicate the detection via the Neo-pixel LED and the buzzer.

✦ Expected Output

- https://youtu.be/SCdrDc_XSSc

✦ Description

- The Magicbit Tiny Fire Detector project enables the creation of a sensor that responds to the presence of fire. By utilizing two infrared sensors, a buzzer, and an RGB LED, this project offers a visual and audible alarm system based on the severity of the detected fire.

✦ Components

- IR Sensor 1 (Pin: A5)
- IR Sensor 2 (Pin: A6)
- Buzzer (Pin: 13)
- RGB LED (1 Neo-pixel, Pin: 0)

✦ How it works:

1. **Proximity IR Sensor**
   - Two IR sensors are connected to the Magicbit Tiny board (IR sensor 1 on pin A5 and IR sensor 2 on pin A6).
   - The sensors read analog values based on the intensity of infrared radiation emitted by the fire.

2. **Proximity IR Sensor**
   - The RGB LED (Neo-pixel) provides visual feedback.
   - The LED changes color based on predefined fire severity levels:
   - Green for no fire
   - Yellow for a small fire
   - Red for a considerably large fire
   - If either IR sensor detects a fire, the RGB LED changes color accordingly.
3. **Buzzer Activation**
   - The buzzer is triggered to produce different tones based on the fire severity.

✦ Steps for the Program

STEP 01 Define the libraries, pins and variables

```
#include <Adafruit_NeoPixel.h>

#define NUMPIXELS 1
#define PIN 0

const int irSensor1Pin = A5; // (A6 - Right A5 - left)
const int irSensor2Pin = A6;
const int buzzerPin = 13;
int t;

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
```

STEP 02 Make the set up

```
void setup() {
  Serial.begin(9600);
  pixels.begin();
  t = 0;
}
```

STEP 03 Make the loop

```
void loop() {
  int sensor1Value = analogRead(irSensor1Pin);
  Serial.print("sensorLeftValue: ");
  Serial.print(sensor1Value);
  Serial.print(" ");
  int sensor2Value = analogRead(irSensor2Pin);
  Serial.print("sensorRightValue: ");
  Serial.println(sensor2Value);

  pixels.setBrightness(255);

  if (sensor1Value < 300 || sensor2Value < 300) {
    pixels.setPixelColor(0, pixels.Color(255, 0, 0)); // Red
    Serial.println( " Color : Red");
    sironON(5); // Specify the duration here
  } else if (sensor1Value < 500 || sensor2Value < 500) {
    pixels.setPixelColor(0, pixels.Color(255, 150, 0)); // Orange
    Serial.println( " Color : Orange");
    sironON(20); // Specify the duration here
  } else {
    pixels.setPixelColor(0, pixels.Color(0, 255, 0)); // Green
    Serial.println( " Color : Green");
    sironOFF(); // Specify the duration here
  }
  // Update NeoPixel display
  pixels.show();
  // Delay for stability
  delay(50);
}
```

```
void sironON(int t) {
  for (int i = 2000; i <= 3500; i += 100) {
    tone(buzzerPin, i);
    delay(t);
  }
  for (int i = 3500; i >= 2000; i -= 100) {
    tone(buzzerPin, i);
    delay(t);
  }
}
void sironOFF() {
  noTone(buzzerPin);
  delay(10);
}
```

Full Program

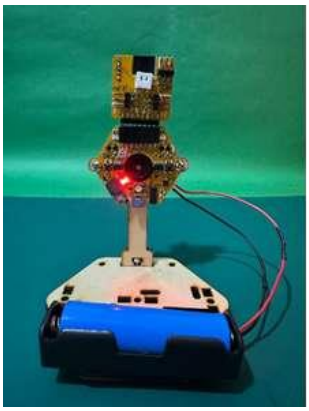
[Download the program code from here]

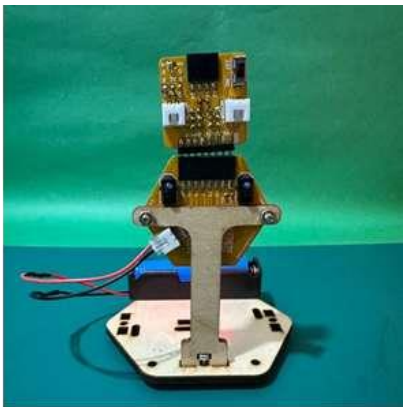
Follow below steps to upload example code.

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, go to **tool** tab and select **Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. Go to **file -> New Sketch** and copy and paste above Arduino code to your new sketch and upload the code.

✦ Set-Up:

- Fix the Magicbit Tiny into the wooden stand as follows.
- Use a candle or a lighter as the fire





Learning Points:

- IR Sensor Sensing: Understanding how infrared sensors detect fire based on emitted radiation.
- Visual and Audible Indication: Implementing an RGB LED and buzzer for effective feedback.
- Experimentation: Customizing thresholds and tones for different fires severity levels.

- Adjust the IR sensor sensitivity by modifying the threshold value.
- Experience with different RGB LED colors and buzzer patterns for enhanced visual and audible alerts.

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/U7jE-M-VZIc

19. DIY Color Sensor

ACTIVITY 19

To create a program to detect the ambient light's color using LDR and displays it through a Neo-Pixel RGB LED.

✦ Expected Output

- https://youtu.be/KZLFKqs3maE

✦ Description

- The Magicbit Tiny Color Sensor project uncover the magic of detecting and visualizing colors using the Magicbit Tiny board, an LDR (Light Dependent Resistor), and a Neo-Pixel RGB LED. This project introduces you to the wonders of color detection and provides a captivating visual display of the colors around you. Witness the colors of your surroundings come alive in a dazzling display of light.

✦ How it works:

1. **Color Sensing with LDR:**
   - The LDR is used to sense the ambient light's intensity and color.
   - The LDR's resistance changes based on the color of light falling on it.
2. **Neo-Pixel RGB LED Display:**
   - The Neo-Pixel RGB LED visually represents the detected color.
   - Each color (Red, Green, Blue) is associated with specific conditions detected by the LDR.
   - The Neo-Pixel lights up in the corresponding color based on the color detected.

Steps for the Program

STEP 01 Define the libraries, pins and variables

```
#include <Adafruit_NeoPixel.h>
#define NUMPIXELS 1
#define RGB_PIN 0
#define LDR_PIN A7

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, RGB_PIN, NEO_GRB + NEO_KHZ800);
```

STEP 02 Make the set up

```
void setup() {
  pixels.begin();
}
```

STEP 03 Make the loop

```
void loop() {
  int ldrValue = analogRead(LDR_PIN);

  if (ldrValue > 500 && ldrValue < 700) {
    pixels.setPixelColor(0, pixels.Color(255, 0, 0));  // Red color
  } else if (ldrValue > 800 && ldrValue < 900) {
    pixels.setPixelColor(0, pixels.Color(0, 255, 0));  // Green color
  } else if (ldrValue > 900 && ldrValue < 1000) {
    pixels.setPixelColor(0, pixels.Color(0, 0, 255));  // Blue color
  }
  else pixels.setPixelColor(0, pixels.Color(255, 255, 255)); // (White Color color)

  pixels.show();

  delay(100);
}
```

Full Program

[Download the program code from here]

Follow below steps to upload example code.

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, Go to **tool** tab and select **Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. Go to **file -> New Sketch** and copy and paste above Arduino code to your new sketch and upload the code.

- Place the Magicbit Tiny Color Sensor (LDR) in different lighting conditions and observe the Neo-Pixel displaying the corresponding color based on the ambient light.
- Experiment with different threshold values for Red, Green, and Blue to fine-tune color detection.
- Try adding more Neo-Pixels to create a colorful light display based on the dominant color detected.

Learning Points:

- LDR Color Sensing: Understanding how LDRs detect colors based on changes in resistance.
- Visual Indication: Implementing a Neo-Pixel for an engaging and colorful visual display.
- Experimentation: Adjusting LDR thresholds to customize color detection.

20. Automatic Plant Watering System

To create a program to water the plant automatically in given time intervals.

✦ Expected Output

- https://youtu.be/JVMcvTRdU14

✦ Description

- The Automatic Plant Watering Project using the Magicbit Tiny Board and a Servo Motor transforms traditional plant care by introducing an efficient, hands-free watering mechanism. The servo motor controls the water flow, providing a simple and effective solution for plant enthusiasts.

✦ Components:

1. Magicbit Tiny Board
2. Tiny Extension board
3. SG90 Servo Motor
4. Wooden parts from Magicbit Tiny Kit
5. Connecting Wires

✦ How it works:

1. **Servo Motor Control:**
   - The Magicbit Tiny board interfaces with a servo motor to control the opening and closing of the water valve. Servo motor can do the blocking and unblocking the water flow through the tube.
2. **Tap Opening and Closing:**
   - The `**openTap()**` function activates the servo to open the tap, allowing water to flow.
   - The `**closeTap()**` function closes the tap, stopping the water flow.

3. **Automated Watering Cycle:**
   - In the `**loop()**` function, the tap opens for a specified duration to water the plant.
   - After watering, the tap closes, and the system waits before checking the moisture level again.

✦ Set-Up:

- Make the set-up as the steps given below.
- Set up Assembly guide - https://youtu.be/UXcOkR2513Q

✦ Steps for the Program

STEP 01 Define the libraries, pins and variables

```
#include <TinyServo.h>

Servo myservo;  // create servo object to control a servo
int servoPin = 8;  // Servo signal pin connected to digital pin 8
int openAngle = 30;  // Angle to open the tap
int closeAngle = 150;  // Angle to close the tap
```

STEP 02 Make the setup

```
void setup() {
  myservo.attach(servoPin);  // attaches the servo on pin 4 to the servo object
  Serial.begin(9600);
}
```

STEP 03 Make the **"open"** and **"close"** functions

```
void openTap() {
  myservo.write(openAngle);
  delay(1000);  // Adjust this delay as needed to allow time for water flow
}

void closeTap() {
  myservo.write(closeAngle);
  delay(1000);  // Adjust this delay as needed to allow time for water flow to stop
}
```

STEP 04 Make the loop

```
void loop() {
  openTap();
  delay(6000);  // Adjust this delay as needed to water the plant for an appropriate duration
  closeTap();
  delay(4000);  // Waiting before checking moisture level again. Adjust the time
}
```

SPECIAL NOTE:

When you are dealing with servo motor with Magicbit tiny you have to use **TinyServo.h** library instead of **Servo.h** library. It will be automatically downloaded when to install the Magicbit Tiny board via Arduino board manager.

Full Program

Download the program code from here

Follow below steps to upload example code.

Follow the steps below to upload example code for path clearing robot car project.

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, Go to **tool** tab and select **Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. Go to **file -> New Sketch** and copy and paste above Arduino code to your new sketch and upload the code.

Learning Points:

- -Servo Motor Control: Gain insights into using a servo motor for precise control in automation projects.
- Timing and Delays: Understand the importance of timing and delays in coordinating actions, such as opening and closing a water tap.
- Automated Systems: Explore the concept of automation in daily tasks, emphasizing efficiency and convenience.

Benefits and Future Enhancements:

1. Water Efficiency: Prevent overwatering by customizing the watering duration.
2. Remote Monitoring: Integrate sensors to monitor soil moisture remotely.
3. Solar Power: Implement solar-powered solutions for sustainable plant care.

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/woOrB6Y_ZSo

21. Automatic Water Tap

To create a program to operate the tap automatically with the presence of hands.

✦ Expected Output

- https://youtu.be/aVyoYc11diQ

✦ Description

- The Automatic Plant Watering Project using the Magicbit Tiny Board and a Servo Motor transforms traditional plant care by introducing an efficient, hands-free watering mechanism. The servo motor controls the water flow, providing a simple and effective solution for plant enthusiasts.

✦ Components:

1. Magicbit Tiny Board & Extension Board
2. Arduino Magic Kit (Automatic Water Tap Kit Setup)
3. Servo Motor
4. Battery

✦ How it works:

1. **IR Sensor Detection:**
   - IR sensors detect the presence of an object (such as a hand) near the tap.
   - Sensor readings indicate whether an object is within the detection range.
2. **Servo Motor Control:**
   - The servo motor regulates the water tap's opening and closing mechanism.
   - Upon detecting an object, the servo motor opens the tap to allow water flow.
   - After a set duration, the servo motor closes the tap to stop water flow.

✦ Set-Up:

- Make the set-up as the steps given below.
- Set up Assembly guide - https://youtu.be/eGumruqGdo4

✦ Steps for the Program

STEP 01 Define the libraries, pins and variables

```
#include <TinyServo.h>
Servo myservo;  // create servo object to control a servo
int irSensor1 = A5;  // IR sensor 1 connected to analog pin A5
int irSensor2 = A6;  // IR sensor 2 connected to analog pin A6
int irThreshold = 980;  // Adjust this threshold value based on your IR sensor readings
int a = 150;
int b = 40;
```

STEP 02 Make the setup

```
void setup() {
  myservo.attach(8);  // attaches the servo on pin 8 to the servo object
  Serial.begin(9600);
}
```

STEP 03 Make the **"open"** and **"close"** functions

```
void washHands(int a) {
  for (int pos = a; pos >= 40; pos -= 1) {
    myservo.write(pos);
    delay(7);
  }
  delay(2000);  // Wait for 2 seconds with the hands under the water
}

void Tapclose(int b) {
  for (int pos = b; pos <= 150; pos += 1) {
    myservo.write(pos);
    delay(7);
  }
}
```

STEP 04 Make the Loop

```
void loop() {
  int sensor1Value = analogRead(irSensor1);
  int sensor2Value = analogRead(irSensor2);
  Serial.print("sensor1Value : ");
  Serial.print(sensor1Value);
  Serial.print("  , sensor2Value : ");
  Serial.println(sensor2Value);
// If both sensors detect an object (hand), trigger the servo
  if (sensor1Value < irThreshold || sensor2Value < irThreshold) {
    washHands(a);
    a = 40;
    b = 40;
  } else if (sensor1Value > irThreshold && sensor2Value > irThreshold) {
    Tapclose(b);
    a = 150;
    b = 150;
  }
}
```

Full Program

Download the program code from here

SPECIAL NOTE:

When you are dealing with servo motor with Magicbit tiny you have to use **TinyServo.h** library instead of **Servo.h** library. It will be automatically downloaded when installing the Magicbit Tiny board via Arduino board manager.

Follow the steps below to upload example code for Roach Robot project.

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, go to **tool** tab and select **Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. Go to **file -> New Sketch** and copy and paste above Arduino code to your new sketch and upload the code.

- Experiment with different servo motor positions and delays to optimize water flow control.
- Explore alternative sensor technologies for object detection, such as ultrasonic or capacitive sensors.
- Customize the project by incorporating additional features, such as temperature sensing or water level monitoring.

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala – https://youtu.be/_uReyCEsHfU

22. DIY Radar System

To create a program to create a radar system capable of detecting objects within a limited range.

- Expected Output
  - https://youtu.be/FIk1webzWMw
- Description
  - Delve into the world of object detection and visualization with the Magicbit Tiny Radar Project, crafted with the Arduino Magic Kit. This exciting endeavor combines the power of Arduino and Processing IDE to create a radar system capable of detecting objects within a limited range. Let's embark on this journey together and explore the magic of technology. The Magicbit Tiny Radar Project combines hardware and software to create a radar-like system capable of detecting objects within a certain range. By utilizing ultrasonic sensors and visualization techniques in Processing IDE, this project offers a glimpse into the world of object detection and tracking.

- Components:
  1. Magicbit Tiny Board
  2. Ultrasonic Sensor
  3. Connecting Wires
  4. Servo Motor
  5. Arduino Magic Kit (Radar Project Kit)

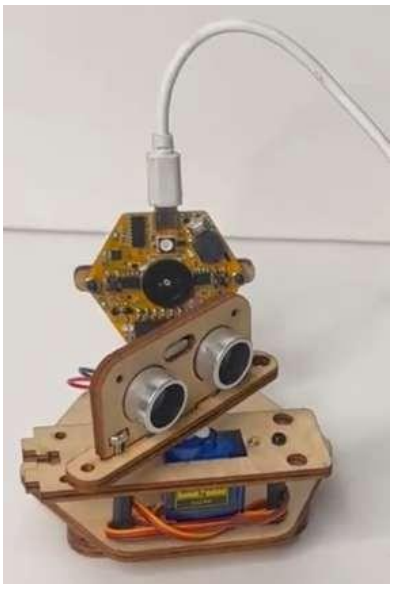
- Software required to install:
  - Processing ID – Download this file

✦ How it works:

1. **Ultrasonic Sensor Integration:**
   - The Magicbit Tiny board interfaces with an ultrasonic sensor to measure the distance between the sensor and nearby objects.
   - By emitting ultrasonic waves and analyzing the echo, the system determines the distance of the detected object.
2. **Data Visualization:**
   - The Processing IDE translates the distance data received from the Magicbit Tiny board into visual representations.
   - Through graphical elements, such as lines and shapes, the system illustrates the presence and position of objects detected by the radar.
3. **Real-time Monitoring:**
   - As the Magicbit Tiny Radar operates, it continuously updates the displayed information, providing users with real-time insights into their surroundings.

✦ Set-Up:

- Make the set-up as the steps given below.
- Set up Assembly guide https://youtu.be/MNS3io9HoHo



✦ Steps for the Program

Download the Arduino Code Here

Important - Install the "**Ultrasonic.h**" library to your Arduino interface before you run the below program. The library used for ultrasonic sensor with Magicbit Tiny is a custom made library.

Download the Ultrasonic Library Here

Upload this code to the Magicbit Tiny board

✦ Processing Code

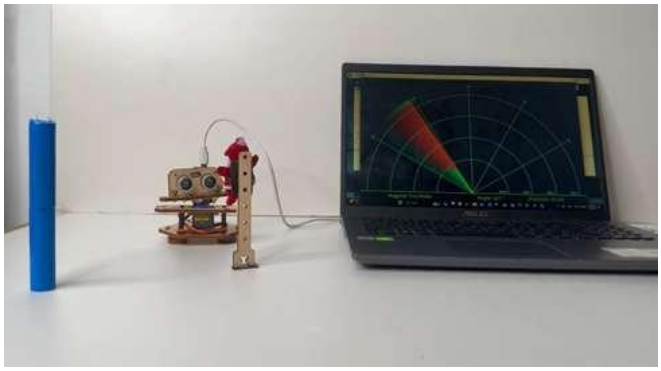
- Open the processing IDE
- Download the below code and copy the code and paste it in the processing IDE

  Click here to download the Code

Note - Before uploading the processing code find out the USB port which Magicbit tiny board is connected using Arduino IDE and edit the processing code by including that.

myPort = new Serial(this,"COM6", 9600); // starts the serial communication

Place objects near to the setup and see the radar on the display.



Learning Points:

- Sensor Integration: Gain insights into interfacing ultrasonic sensors with microcontrollers for distance measurement.
- Data Visualization: Explore techniques for visualizing sensor data in real-time using Processing IDE.
- Interactive Systems: Understand how to create interactive systems that respond to changes in sensor inputs.

If you need the same activity in Sinhala Medium, go through the below Video.

Tutorial in Sinhala - https://youtu.be/evV3IPwXGhE

23. Bluetooth Remote Controlling Car

Create a program to control simple robotic car via Bluetooth

✦ Expected Output

- https://youtu.be/bcxtcEXkuvE

✦ Description

- Welcome to the exciting world of the Magicbit Tiny Bluetooth Remote Control Car project! Get ready to explore how the Magicbit Tiny board can transform into a fun car controlled via Bluetooth. Let's dive into the simple documentation and unveil the secrets of this thrilling project. The Magicbit Tiny Bluetooth Car project enables you to create a remote-controlled car using the Magicbit Tiny board and a Bluetooth module. With this project, you can send commands from your smartphone to control the car's movements.

✦ Components:

1. Magicbit Tiny With Extension
2. Bluetooth Module (HC-05 or HC-06)
3. In-Built Motor Driver Modules in Magicbit Tiny Board
4. Motors (2x)
5. Chassis and Wheels
6. Battery
7. Mobile Phone with "Arduino Car" Application.

✦ Arduino Car Application

- Click here to download Arduino Car mobile app

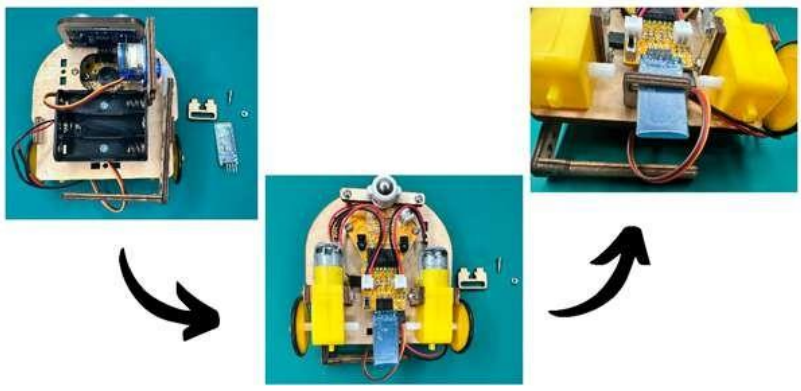
✦ How it works:

1. **Bluetooth Communication:**
   - Bluetooth module is integrated with the Magicbit Tiny board to enable wireless communication. The Bluetooth module receives commands from the paired device.
2. **Motor Control:**
   - The project uses two in-built motor driver modules in Magicbit Tiny to control the direction of rotation of the motors. By activating specific motor pins, we can make the car move forward, backward, left, right, or stop.

✦ Set-Up:

- Make the set-up as the steps given below.
- Set up Assembly guide - https://youtu.be/q6y0NJX4XmA

- Connect the **"Bluetooth Module"** to the above car structure as follows.



✦ Program Code

- Download Bluetooth Car Code

Follow the steps below to upload example code for Roach Robot project.

1. Connect Magicbit Tiny using a USB cable to a computer.
2. Open the Arduino IDE, go to **tool** tab and select **Board -> Magicbit Tiny -> MagicBit Tiny**
3. In tool tab select **port -> COM<your port number>**
4. Go to **file -> New Sketch** and copy and paste above Arduino code to your new sketch and upload the code.

- Pair your smartphone with the Bluetooth module. (Appeared name :- Tiny Car)
- Open a Bluetooth terminal app on your smartphone.
- Configure the buttons of the app (F' for forward, 'B' for backward, 'L' for left, R' for right, 'S' for stop).
- Send commands
    a. 'F' for forward
    b. 'B' for backward
    c. 'L' for left
    d. 'R' for right
    e. 'S' for stop from the app to control the car.

Learning Points:

- Bluetooth Communication: Understanding wireless communication using Bluetooth.
- Motor Control: Learning how to control motors for various movements.
- Remote Control: Exploring the concept of remote-controlled devices.

You've now unlocked the power of Magicbit Tiny and Arduino!

We hope this guide has inspired you to explore the endless possibilities of electronics and programming. But this is just the beginning.