Thunkable is a web-based programming system developed by Rappidly Inc. that transforms the complex language of text-based coding into visual, puzzle piece-shaped building blocks. It is provided free of charge.

Arduino is an open-sourced electronics prototyping platform with easy to use hardware and software suitable for makers and tinkerers of all ages. The Arduino name and logo are trademarks of Arduino. Circuits shown in this manual is produced with TinkerCAD, an open-sourced circuit simulator provided by Autodesk. The TinkerCAD name and logo are trademarks of Autodesk.

The information in this manual has been obtained from sources believed to be reliable at the time of publication. The author does not guarantee the accuracy and the completeness of information presented herein, and shall not be responsible for any errors, omissions or damages as a result of the use of this information. No part of this manual may be reproduced, copied, or distributed in any form or by any means without the express written permission from the author.
Hi! Congratulations for being one of the many young engineers selected to participate in this year’s IET Faraday Challenge Malaysia!

This handout contains all the information you need to prepare for the big day! If you’ve already had a MakerUno kit from Cytron, great! Or else, you can choose to purchase one here: https://www.cytron.io/p-maker-uno-edu-kit-arduino-compatible. Either way, the MakerUno kits will be provided by the IET during the competition day, so it is not compulsory to purchase one.

This training manual will start by introducing TinkerCAD, a very useful Arduino Simulator, which allows you to build circuit, and simulate it right from your browser! It allows you to either code from scratch, or to use a block-based interface.

The second part of the training manual will focus on Thunkable, a block-based mobile app development software, which allows you to run and build Android or iOS apps right from your browser. Do remember to seek permission from your school teachers before bringing your phone to school!

During the competition day, our super helpful technical managers will be on site to provide help! We will also teach you how to connect your apps and the Arduino through Bluetooth.
IN THE BOX

**Maker Uno Edu Kit**
The Most Affordable Arduino Kit to Kickstart Your Arduino Class

1. Maker UNO x 1
2. USB Micro B Cable x 1
3. LED 5mm Red x 3
4. LED 5mm Yellow x 3
5. LED 5mm Green x 3
6. Breadboard (Small) x 1
7. 40 Ways Male to Male Jumper Wire x 1
8. Resistor 0.25W 5% (220Ω) x 10
9. Resistor 0.25W 5% (1K) x 5
10. Resistor 0.25W 5% (10K) x 5
11. LDR (Small) x 1
12. Transistor 2N2222 x 1
13. Finger Adjust Preset 10K x 1
14. 3V Miniature Brush Motor w/o Gear x 1
15. DIY 4 Blades 56mm Motor Propeller x 1
16. Plastic Box x 1
THE SCENARIO...

As computers become more common in our everyday lives engineers are using coding more frequently to solve everyday problems. Today your challenge is to create a solution for real life problems. You will work in teams of engineers to solve a problem or to change or improve people’s experience in an area of our everyday lives. You will experience what engineers do as they work together to develop a new product. You will need to use all your STEM skills as well as skills in teamwork, perseverance, creativity and innovation.

You will need to be brave — engineering is not for the faint hearted!

DURING THE COMPETITION...

You will be given “Faradays” a virtual currency to be used to buy extra sensors and help from our technical managers. Each team will be provided with a MakerUno. The other items from the Maker Uno Kit will be removed.

A “Shop” will be set up in a corner of the competition hall, where you may send a representative to buy extra sensors needed for your project using “Faradays”. You will need to conduct a cost analysis to ensure that your project can be done within the allocated “budget”. This is to simulate the real-world environment, where engineers have to solve problems within a certain amount of budget allocated to them.

Nearing the end of the competition, your team should prepare for a quick 3 minute pitch, which consists of: (1) Your problem statement, (2) Your solution, (3) costing and budget. The 3 minute pitch will be followed by a 2 minutes quick demo of your project.
PART 1:

INTRODUCTION TO

TINKERCAD

ARDUINO
1.1 USING TINKERCAD

Objective:
- Familiarise students with TinkerCAD’s interface
- To get students comfortable in running electronic simulations before building circuits.

TinkerCAD allows you to run Arduino codes on a simulator, even if you do not own an Arduino, hence it is the perfect tool to start your preparation for the competition!

- Go to https://www.tinkercad.com
- Click on “Join now” if you wish to create an Autodesk account. Alternatively, you can click on “Sign in”, and select “Sign in using social providers” to login with your Facebook/Google/Microsoft accounts.

- Ignore the prompt from TinkerCAD 3D.
If all goes well, you should be able to see the page as shown below:

- Click on “Circuits” to access the circuit simulator.

In the circuit simulator, you have the choice of going through the online tutorial, or to create a new circuit.

For the sake of brevity, we will proceed to create our own circuit. You may explore the tutorial at your own time.
The general structure of the simulator is shown:

If there's a programmable element in the circuit (i.e. Arduino), pressing the “Code” button will toggle the coding area. Clicking “Block” in the coding area will allow you to switch between block based programming and code based programming.
Clicking “Components” in the component library will allow you to switch between standard basic electronic components and Arduino based circuits.

The sample circuits shown here are preloaded with sample codes. For this tutorial, let’s drag the “Button” demo into the main work area.

After dragging the circuit into the work area, click on the “Code” button and observe the codes for this demo:

Ignore the gray “Comment” blocks

Purple blocks are Variable blocks, they are used to store values. In this example, a variable called “buttonState” is being created to store the value from the input on pin 2.

Indigo blocks are Input blocks, they are used to take input from the digital and analog pins of the Arduino.

Orange blocks are Control blocks, which is used to control the logical flow of the program. In this case, it is used to implement an “If” condition, i.e. If button is “High” (which indicates that it is pressed), then the LED is turned on (Turned “High”).

Green blocks are conditional blocks, which is used to check certain conditions, such as “equal”, “larger than”, “smaller than” etc.
To start the simulation, click on the “Start simulation” button

When the simulation is started, the USB cable will be connected to the Arduino. If everything works well, the LED on board will light up when you click on the button.

If you have an Arduino, you can download the codes by clicking on the download button. The downloaded code will be in the form of an “.ino” file, which is compatible with the Arduino IDE. For more information on how to flash the code to an actual Arduino, refer to: https://www.arduino.cc/en/main/howto
1.2 BLINKY BLINKY!

Objective:
- To complete a simple project from scratch
- Introduce the concept of digital IO and delay function

Start a new circuit project in TinkerCAD and create the circuit shown:

Common mistake alert!
Make sure the cathode is connected to ground (GND) and the anode is connected to the pins. LEDs do not work if connected backwards!

The LED colours can be changed by clicking on it, and selecting a different colour in the properties box.

The same can be done to the wires. Changing colours will not affect the circuit, but it makes it easier to understand.

When this is done, change the value of the resistance to 220 Ohm, by clicking on the resistor and changing the “resistance” value. This will prevent the LEDs from burning out.

Extra Info: Why do we use 220 Ohm? Read this forum to find out more: https://forum.arduino.cc/index.php?topic=121606.0
Click on codes and make the following program:

This code basically turns the LEDs on and off with a one second delay.

Do note that the LEDs are connected to pins 2 and 3, hence in the code, pin 2 and pin 3 are set to HIGH and LOW.

Once you are ready, click the “simulate” button. You should be able to see the LEDs blinking.

Challenge: Make alternating LED flashes lasting 2 seconds each

Hint: You only need to do some minor change to the code blocks!
Now, add a button to the circuit

A button connected in this configuration is known as the “Active High” switch.

This means that when the switch is active (pressed), pin 5 will read a “HIGH” (5V). When the button is released, pin 5 will read a “LOW” (0V) due to the pull-down resistor.

The pull-down resistor is always set to an arbitrary large value. In this case, it is set to 1 MΩ.
Add the code to control the LEDs with button

For this, we will need a “If-Else” control block. Remember, Control blocks are always orange in colour.

The condition for the If-Else block can be found in the “Math” tab, while the input can be found in the “Input” tab.

Remember to set the input pin to digital pin 5

What’s the difference between digital pins (0-13) and analog pins (A0-A5)?

Digital pins can only read binary values. Which means they can only be switched ON (5 V) or OFF (0 V).

Analog pins can read voltage range between 0 V and 5 V. Due to the configuration of the Arduino’s ADC (analog to digital converter), The Arduino reads 5 V as 1023, and 0 V as 0. Hence, any values in between can be mapped linearly between 0 and 1023.

Now modify the code so that the LED toggles each time the button is being pressed.

Hint: You will need to use some blocks which are not mentioned in this tutorial.

Try playing around with the different logic functions.
1.3 SWEEPING ARM

Objective:
- To complete a simple project from scratch
- To introduce the concept of PWM and servo control.
- To introduce simple concepts of ADC

1. Start a new circuit project in TinkerCAD, drag the Arduino servo project.

When you click on “Code” after dragging the Servo project to the main area, you will be able to see the following:

2. Try to read the comments (grey blocks) to try and understand what the circuit is doing.

When you are done, click run, and observe the servo.

Is it moving? How is the movement like?
For more information, please watch:
https://www.youtube.com/watch?v=LXURLvga8bQ

This video explains the concept behind a servo, and how it works.

Now, let’s learn how to control a servo using a switch

Connect the circuit and codes as shown.

Essentially, the switch connects either 5v (Red) or Ground (Black) to the common terminal (Green). Hence, pin 11 will read HIGH when the switch is connected to 5V, and LOW when the switch is connected to ground.

For this example, the code is telling the servo to rotate to 90 degrees if the switch is switched to 5V, and rotate to 0 degrees if the switch is switched to ground. Note that there’s always a delay after asking servo to move. This is because the servo will take time to move from one position to another. Hence, a suitable delay must be given to allow the servo to move before a next instruction is given.
Now, let’s learn how to control a servo using a potentiometer

Here, we will learn how to control the servo using a potentiometer (Variable resistor).

Create the circuit as shown in the diagram. The potentiometer can be found under the “Basics” tab in the components panel.

Make sure that the signal line (Green) is connected to an analog input (A0-A5) instead of a digital input.

When you are done, connect the code blocks as shown:

Run the code, and turn the potentiometer by clicking and dragging.

The analog pins of the Arduino reads the input voltage between 0v to 5v and maps it to a analog-digital conversion (ADC) value between 0 and 1023. To map this value to the 0-180 degrees input of the servo, we will need to use the following formula:

\[
CurrentDegree = ADCReading \times \frac{MaxDegree}{MaxADC + 1}
\]
In our case, since the maximum angle is 180, and the maximum ADC value is 1023, So the conversion becomes:

\[ \text{CurrentDegree} = \text{ADCR}eadi\text{ng} \times \frac{180^\circ}{1024} \]

\[ \text{CurrentDegree} = \text{ADCR}eadi\text{ng} \times 0.176 \]

Which is why there is a constant of 0.176 defined in the code. If you wish the servo to rotate between 0 degree and 90 degree, simply replace MaxDegree with 90 instead of 180. Try it out and see for yourself!
1.4 MORE ADC FUN

Objective:
- To reinforce the concept of ADC
- To introduce simple sensors, and sensor integration.
- To introduce the concept of Serial communication and data visualization
- To familiarize students with breadboard prototyping.

1. Start a new circuit project in TinkerCAD, and create the circuits shown below.

In a breadboard, the holes are connected as shown in the diagram above.

This is essentially a voltage divider circuit, where the voltage output \( V_{\text{out}} \) is proportional to the ratio of the two resistance.

This is an important circuit configuration, as it allows us detect a change in electrical resistance as a change in voltage, which can be measured by the Arduino.

\[
V_{\text{out}} = V_{\text{in}} \times \left( \frac{R_2}{R_1 + R_2} \right)
\]

For the photoresistor, it is important to select a correct R1 value. In our case, Vin is 5V, and we want Vout to be close to 5V when R2 is minimum, and close to 0V when R2 is maximum.

In our case, the photo resistor is 20 Ohm when bright, and 175 kOhm when dark. Hence, we selected a \textbf{10 kOhm} reference resistor. Remember to change the value of the resistor in your circuit to 10 kOhm.
2. Write the code as shown:

```
print to serial monitor  read analog pin  A0  with  newline
```

Remember, **blue** blocks can be found in Output, while **purple** blocks can be found in input.

3. Press simulate, and activate the serial monitor to see the output.

You should be able to see the output as shown. Click on the graph symbol to display a graphical view of the input.

Click on the light dependent resistor (LDR) and slide the bar to simulate different light intensity. Observe what happens on the graph.
1.5 NIGHT LIGHT

Objective:
- To reinforce the concept of ADC
- To introduce the concept of analogWrite() and PWM.
- To familiarize students with breadboard prototyping.

Remember the concept of PWM in section 1.3?

Servos use PWMs as signals. PWM can also be used to vary the amount of power supplied to an electronic component, for example, an LED!

Modify the circuit to include an LED. Don’t forget to include a 220 Ohm current limiting resistor to avoid burning the LED!

Modify the circuit from 1.4 to include an LED. Don’t forget to include a 220 Ohm current limiting resistor to avoid burning the LED!

Connect the anode of the LED to any of the PWM-enabled pins (which has the “~” symbol next to the pin number)

When you’re done, it’s time to modify the codes!
Write the code as shown:

The map function automatically maps the value of the ADC (0-1023) to the value of the PWM (0 – 255).

Therefore, it saves you the trouble of converting the numbers manually as we did in section 1.3. However, it is still important for you to learn how the conversion works before using the “map” blocks!

Simulate!

Notice that when the LDR detects dark environment, the LED will light up! And when the LDR detects a bright environment, the LED dims! Now, modify the code so that the LED turns on completely when it is dark enough, and turns off completely if it is bright enough using what you’ve learned earlier!
PART 2:

INTRODUCTION TO thunkable
2.1 USING THUNKABLE

Objective:
- Let students explore the Thunkable app creator
- Let students build their first Android mobile app and test it.

Thunkable allows you to create powerful mobile apps with AI capabilities!

- Go to http://app.thunkable.com

To start a new project, click on the “Create New App” button.

To test your code, make sure you download the Thunkable Classic app, instead of the Thunkable Live app.
2.2 TUTORIALS

Objective:
- Let students create their first Android app
- Teaches the concept of block based programming

Watch videos 1 – 10 and try it out on Thunkable.
2.3 CONNECT TO ARDUINO

Objective:
- Let students link their Android app to Arduino
- To teach the student basics of IoT


Upload the code to your Arduino via the Arduino IDE before connecting the circuit.

Next, start a project in Thunkable as usual, and add the following components to your project:
Go to “Blocks” and create the following blocks:

Here, when screen1 is initialized, the code check if the Bluetooth Client is available. If it is not, then it will show an alert message saying “No Bluetooth”.

This block populates the list with the name and address of the Bluetooth devices available to the Android phone before the user press the “Bluetooth” list picker.

When the user pressed the “Bluetooth” list_picker1 button, the list will be shown. After the user has made the selection, List_picker1.AfterPicking will be activated.
This is the List_Picker1.AfterPicking block. After the user chose the correct Bluetooth device, the Bluetooth Client block was called to connect to the selected device.

After picking, and connecting to the Bluetooth, you can test the connection using one of these two buttons. If your Arduino code was uploaded correctly, your on-board LED should turn on and turn off accordingly.

```
// Faraday Challenge Bluetooth Sample Code
******************************************
String serialInput = " ";
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  // Here, we will read the Serial input only if there is an input. if bluetooth
  // is not connected, we don't read.
  while(Serial.available()) {
    serialInput = Serial.readString(); // read the incoming data as string
  }
  // Here is an example of how we can do something when we receive a command.
  // If your Android App sends an "ON" command, we turn on the LED
  if(serialInput == "ON")
  {
    digitalWrite(LED_BUILTIN, HIGH);
  }
  // If your Android App sends an "OFF" command, we turn off the LED
  else if(serialInput == "OFF")
  {
    digitalWrite(LED_BUILTIN, LOW);
  }
```
# Maker Uno Comparison Table

<table>
<thead>
<tr>
<th>FEATURES</th>
<th>Arduino Uno R3</th>
<th>CT UNO</th>
<th>Maker UNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller</td>
<td>ATmega328P</td>
<td>ATmega328P</td>
<td>ATmega328P</td>
</tr>
<tr>
<td>Operating Voltage</td>
<td>5V</td>
<td>5V</td>
<td>5V</td>
</tr>
<tr>
<td>Input Voltage (recommended)</td>
<td>7 - 12V (Adapter)</td>
<td>7 - 12V (Adapter)</td>
<td>5V (USB only)</td>
</tr>
<tr>
<td>Input Voltage (limit)</td>
<td>5V (USB only)</td>
<td>5V (USB only)</td>
<td>5V (USB only)</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>PWM</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Analog Input</td>
<td>6 (10-bit)</td>
<td>6 (10-bit)</td>
<td>6 (10-bit)</td>
</tr>
<tr>
<td>UART</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SPI</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>I2C</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>External Interrupt</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>DC Current for 5V</td>
<td>1A</td>
<td>1A</td>
<td>USB Source</td>
</tr>
<tr>
<td>DC Current for 3.3V</td>
<td>50 mA</td>
<td>500 mA</td>
<td>500 mA</td>
</tr>
<tr>
<td>DC Current per I/O Pin</td>
<td>20 mA</td>
<td>20 mA</td>
<td>20 mA</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>32 KB</td>
<td>32 KB</td>
<td>32 KB</td>
</tr>
<tr>
<td>SRAM</td>
<td>2 KB</td>
<td>2 KB</td>
<td>2 KB</td>
</tr>
<tr>
<td>EEPROM/Data Flash</td>
<td>1 KB</td>
<td>1 KB</td>
<td>1 KB</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>16 MHz</td>
<td>16 MHz</td>
<td>16 MHz</td>
</tr>
<tr>
<td>USB to Serial Chip</td>
<td>ATmega16U2</td>
<td>FT231X</td>
<td>CH340G</td>
</tr>
<tr>
<td>Programming IDE</td>
<td>Arduino IDE</td>
<td>Arduino IDE</td>
<td>Arduino IDE</td>
</tr>
<tr>
<td>Extra Features</td>
<td>Programmable LED at pin 13</td>
<td>Programmable LED at pin 13</td>
<td>Programmable indicator LED at every digital pins (pin 2 to pin 13)</td>
</tr>
<tr>
<td></td>
<td>Extra GPIO pads with standard pitch</td>
<td>Programmable push button at pin 2</td>
<td>Piezo buzzer at pin 8 with selector switch</td>
</tr>
<tr>
<td>Price</td>
<td>RM 96.00 (~USD 25.00)</td>
<td>RM 58.80 (~USD 15.00)</td>
<td>RM 26.50 (~USD 6.58)</td>
</tr>
</tbody>
</table>

Source: Cytron Technologies Sdn. Bhd.
Arduino PinMap

Source: O'Reilly Media