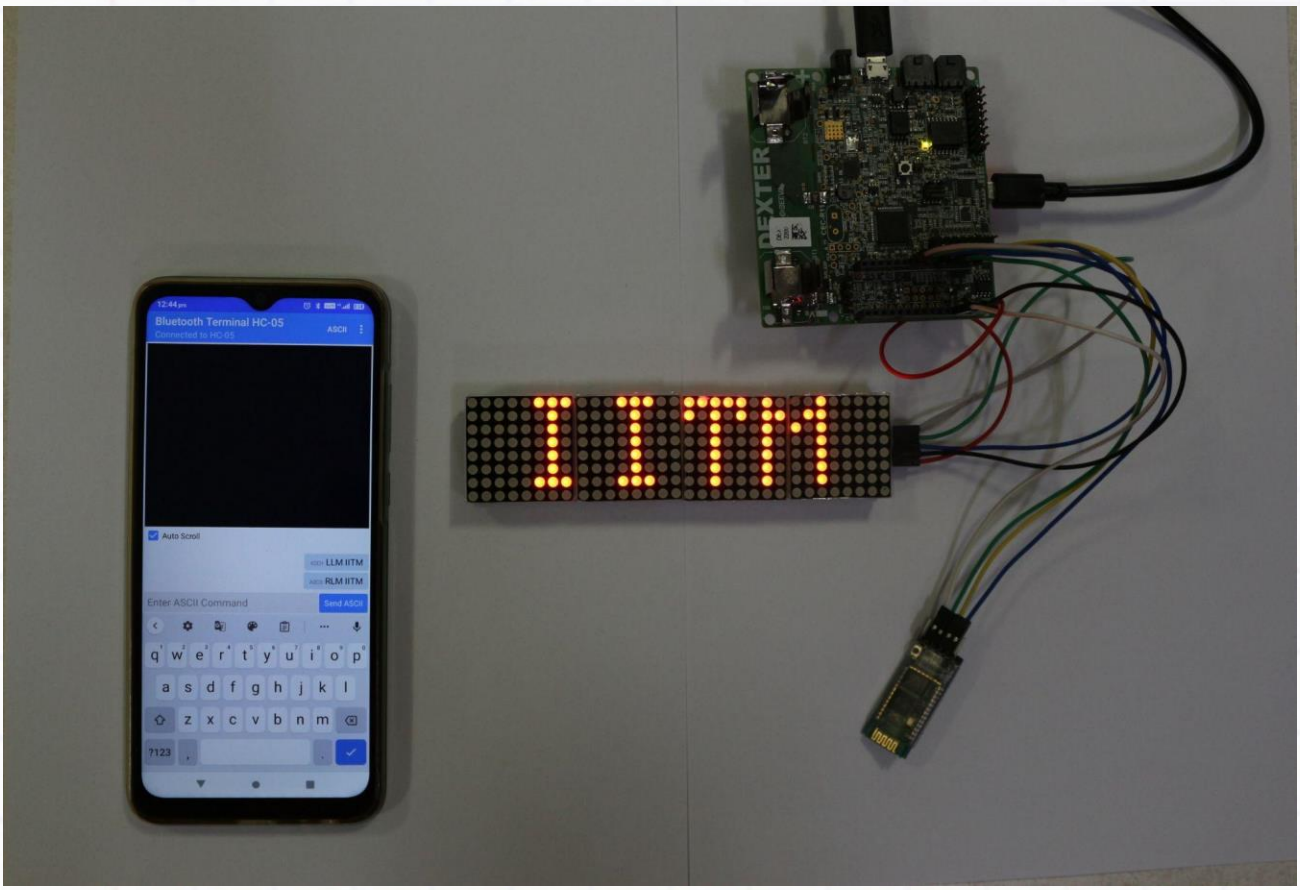
Build a Rolling LED Display!



Contents

Prerequisites

Aim

Components

Connections

- Circuit Diagram
- Detailed Steps

Software

- Launching the IDE for our project
- Code
  - Level 1
  - Level 2

Challenges

Prerequisites

Topic	Resources
Dancing LEDs Project	Dancing LEDs Project
Basic C Programming	C Full course - Youtube
	W3Schools Interactive C Tutorial
	Learn-C.org Interactive Tutorial
Number Systems	Binary, Decimal & Hexadecimal

Aim

Build an LED matrix display and send messages to it wirelessly from your phone using Bluetooth. This includes controlling the Direction and Speed of scrolling as well as Brightness of the display remotely from your phone.

Components (*Provided in kits sent by Build Lab - IITM)

1. Dexter board
2. LED Dot Matrix display - MAX 7219 board with four 8x8 displays mounted.
3. Bluetooth module - HC-05 module
4. Breadboard
5. Jumper wires (9 male to female , 2 male to male)
6. 2 USB cables - 1 for powering the Dexter and the other for uploading code from laptop to Dexter.

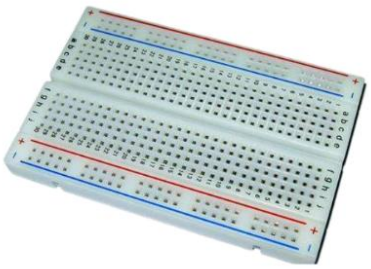
Dexter Board

Dot matrix Display

Bluetooth Module

Jumper Wires

USB

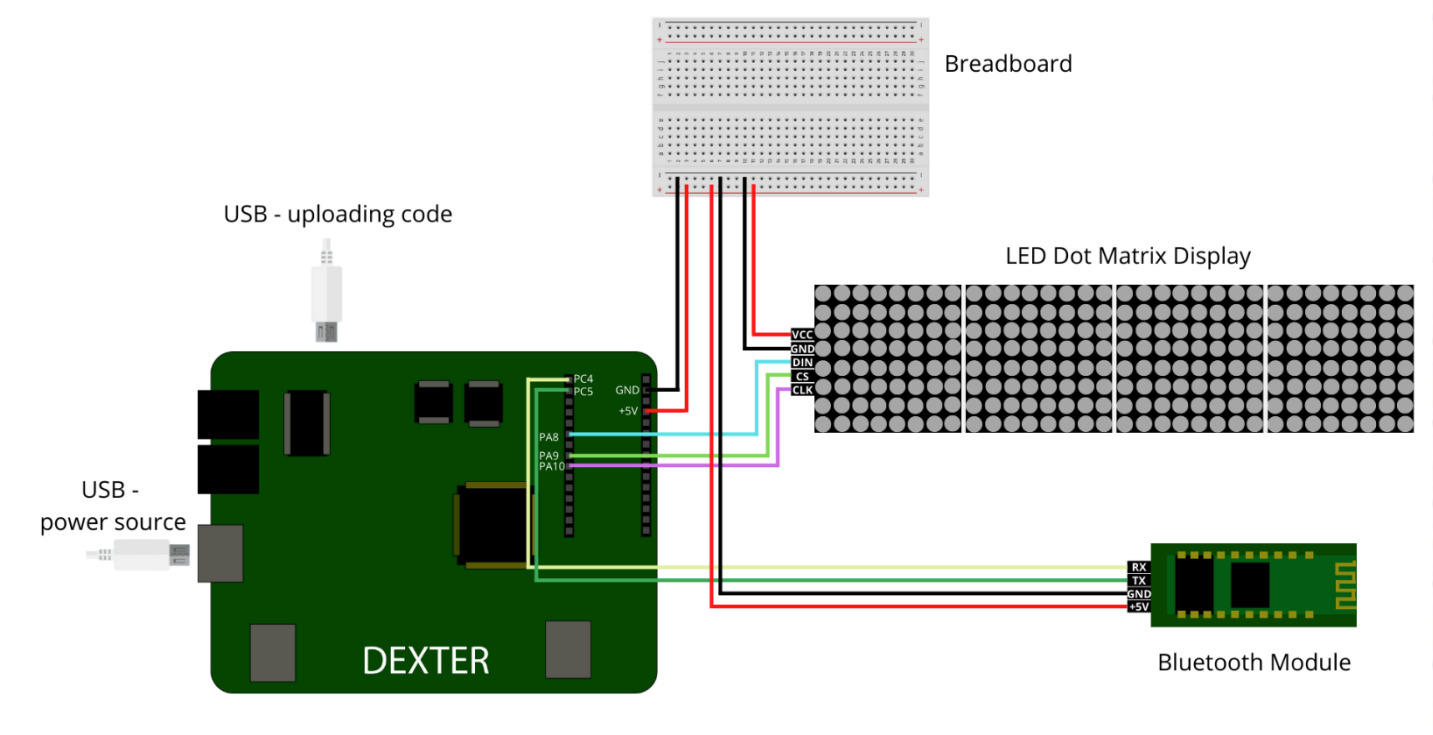


Breadboard

Connections

Safety tip: Always ensure that the connections to the components are correct and completed before connecting the power supply to the Dexter board.

Circuit Diagram



>> Follow the detailed steps in the following pages to complete the circuit.

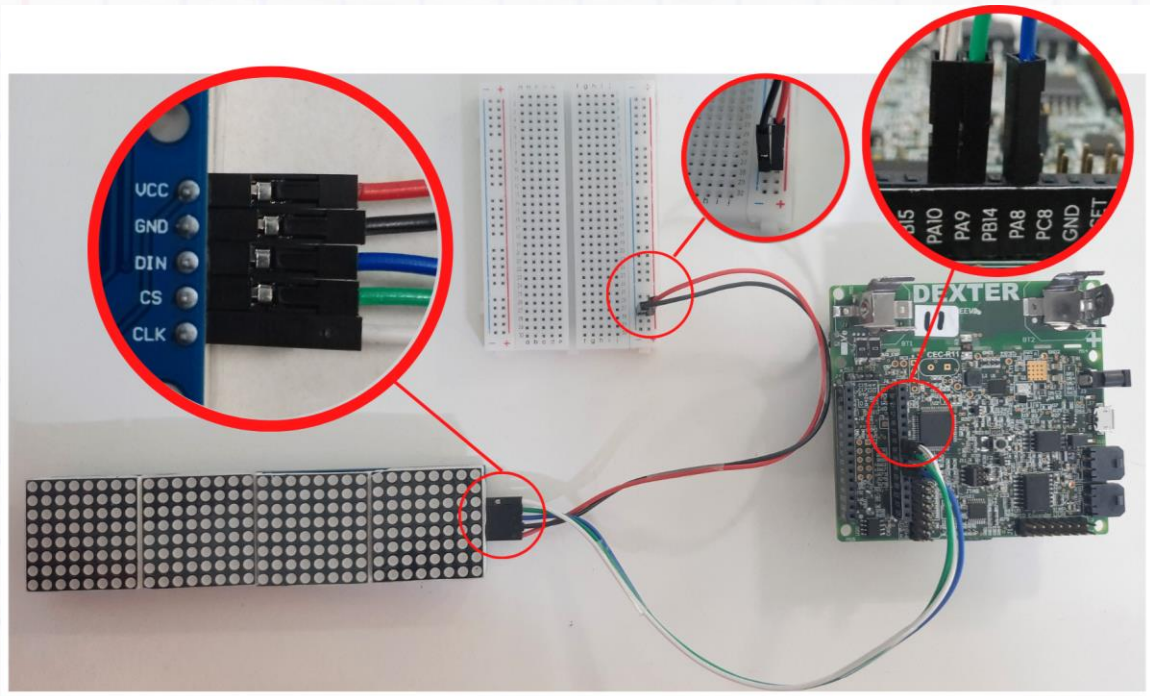
- NOTE: Before starting the connections, verify using a multimeter that all the jumper wires are working. Also ensure that the connections are strong, else the setup may not work.

Detailed Connection Steps

Step 1

Take 5 female-to-male jumper wires and connect them to the pins (VCC, GND, DIN, CS, CLK) of the MAX7219 display board as shown below. Connect the other ends as per the below connections:

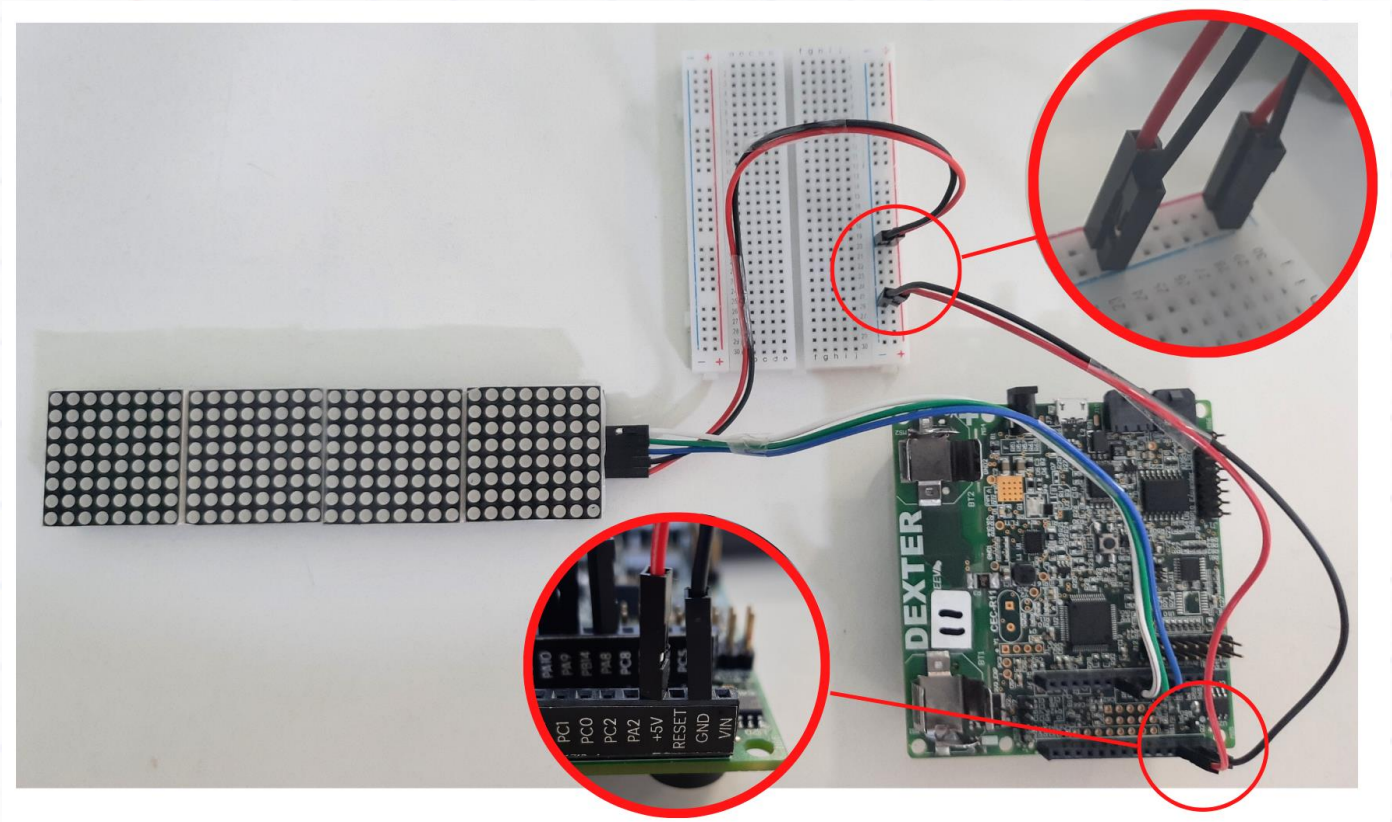
- VCC to Red line of breadboard
- GND to Blue line of breadboard
- DIN to PA8 of Dexter
- CS to PA9 of Dexter
- CLK to PA10 of Dexter



Step 2

Take 2 more **male-to-male** jumper wires and connect them as follows:

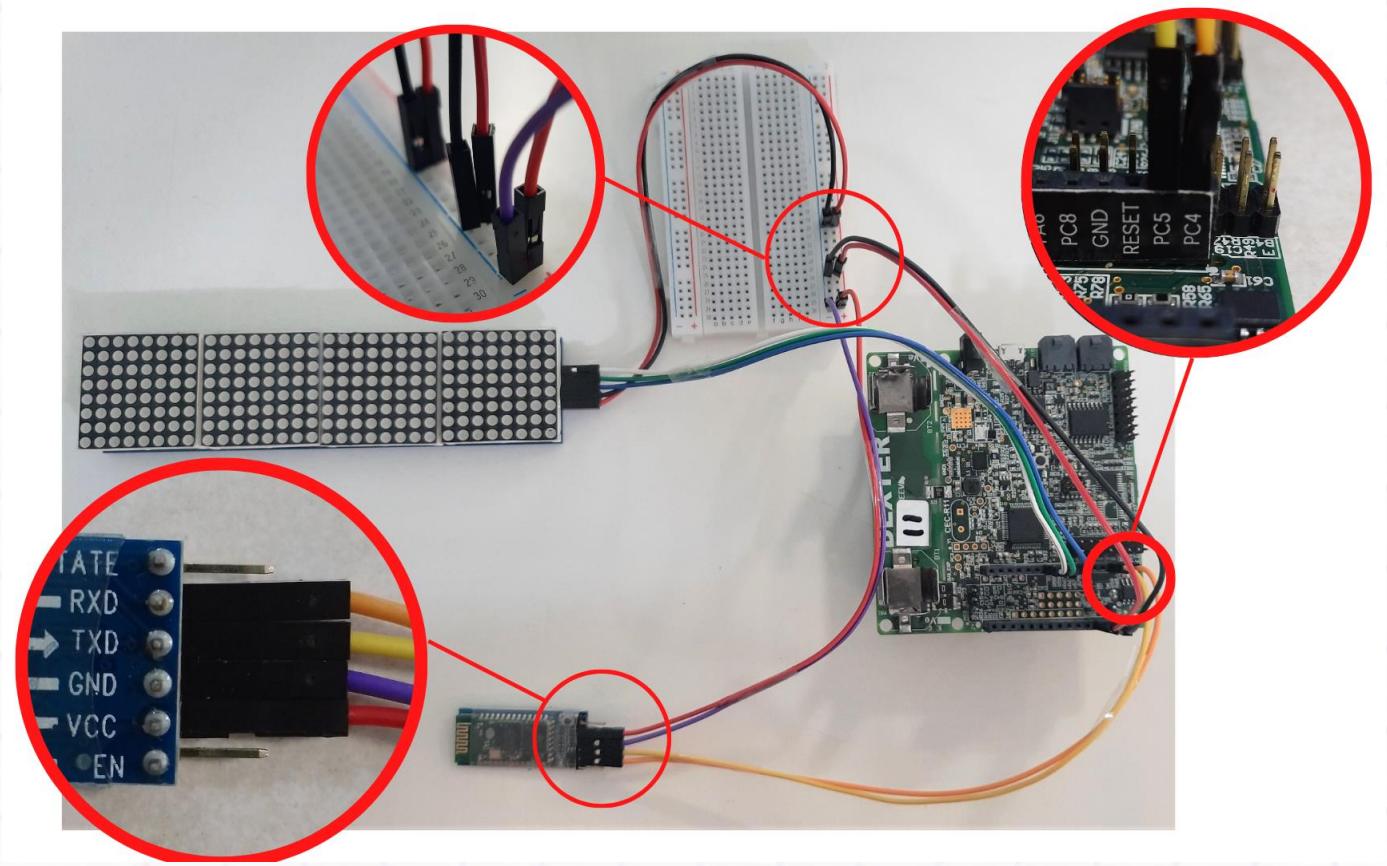
- +5V of Dexter to Red line of Breadboard
- GND of Dexter to Blue line of Breadboard



Step 3

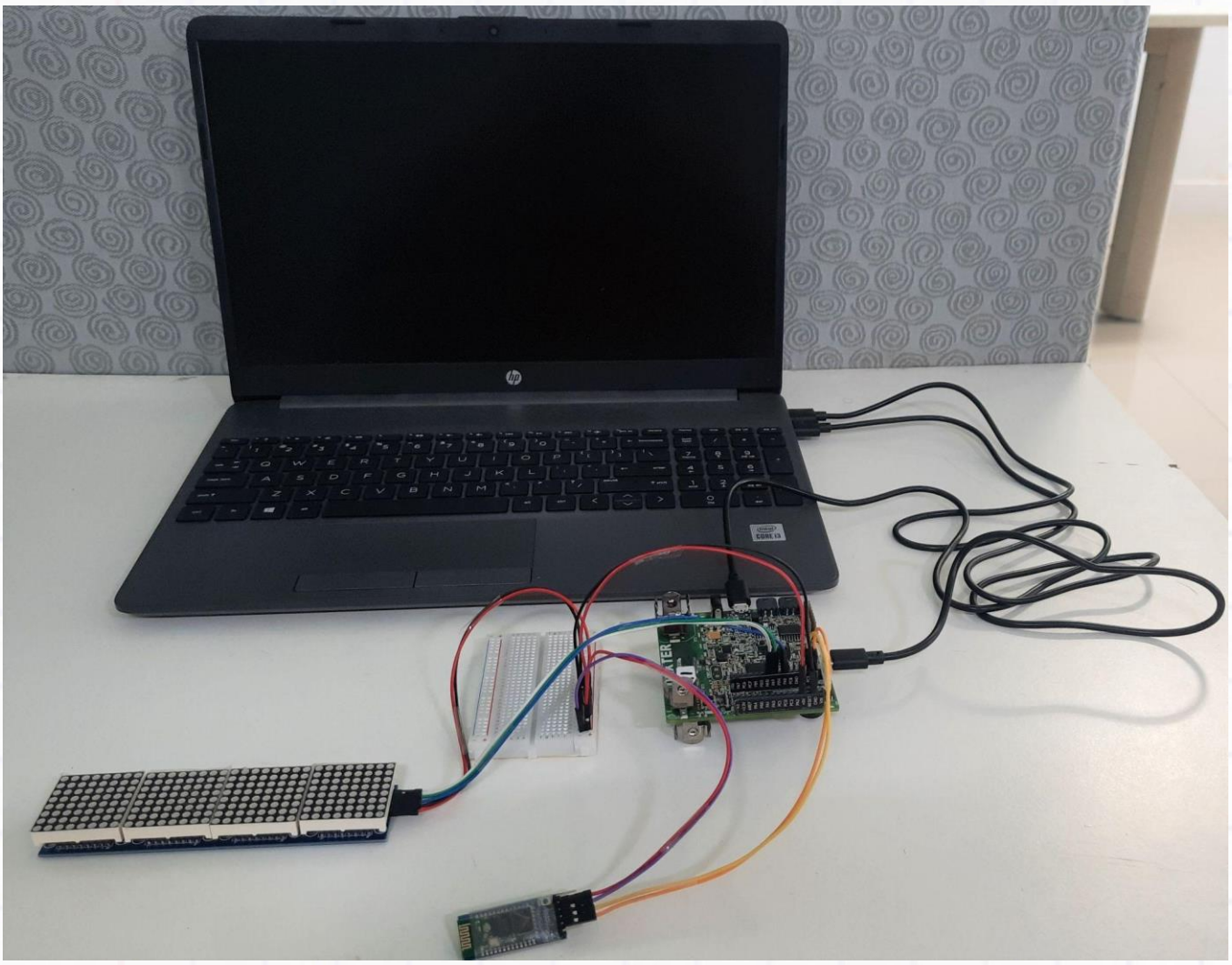
Now take the final 4 **female-to-male** jumper wires and connect them to the pins (+5V, GND, TX, RX) of the HC-05 Bluetooth module as shown below. Connect the other ends as per the below connections:

- RXD to PC4 of Dexter
- TXD to PC5 of Dexter
- GND to Blue line of breadboard
- VCC to Red line of breadboard



Step 4

Connect the USB cables to the Dexter - One for uploading the code and the other for powering the board. The circuit is now complete!



You are now ready to work on the software for the project.

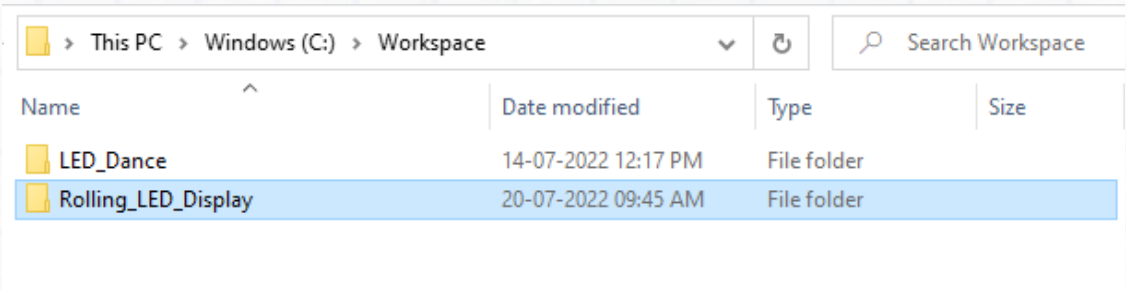
Software

Downloads & Installation

1) To download and install the STM32 Cube IDE (If you haven't already), follow the steps given in the Dexter Base Document, part 2.

2) Next, download the Project Workspace file 'Rolling LED Display.zip' given in the project page on the Build Club website.

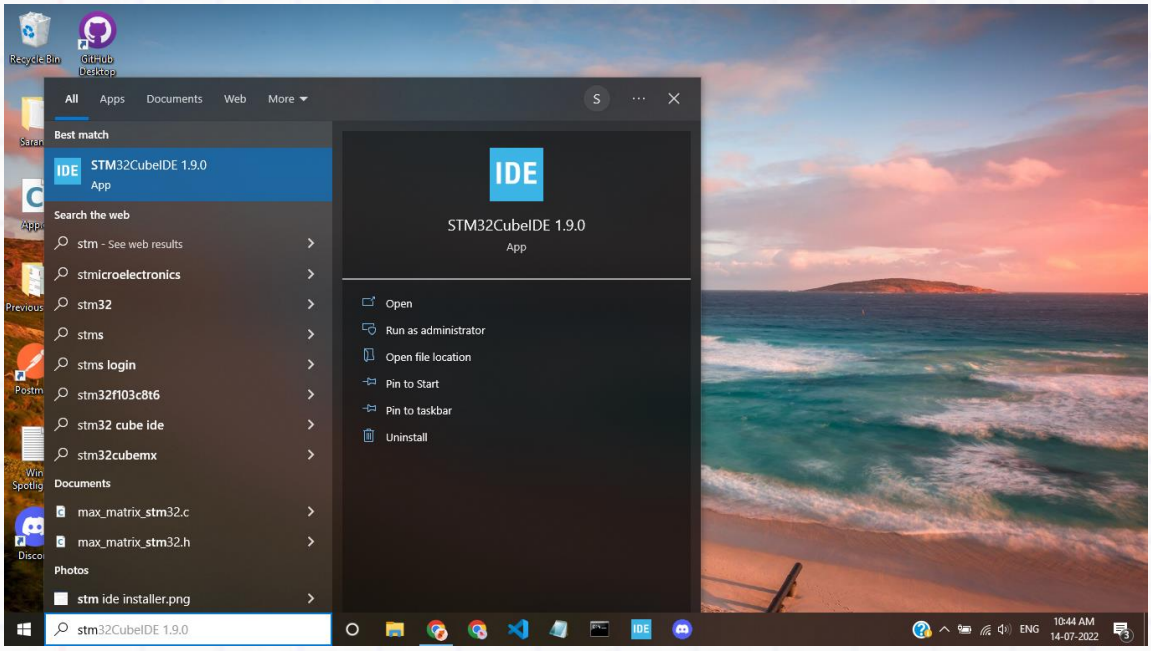
3) In the C: drive on your computer, create a new folder named 'Workspace'. Inside it, create a folder named 'Rolling_LED_Display'.



4) On your phone, download the 'Bluetooth Terminal HC-05' app from Play Store (Link). Open Bluetooth settings on your phone and pair the HC-05 device. Now open the Terminal app & select 'HC-05' from 'Paired Devices'. You will now enter the communication window in the app. The rapid blinking of the HC-05 module when disconnected will now switch to an occasional blinking, once connected to the phone.

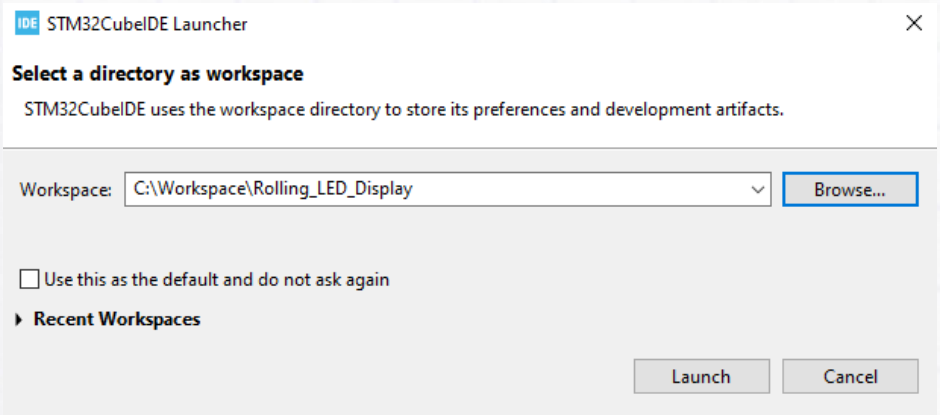
Launching the IDE for our project

Step 1: Launch the STM32 Cube IDE application on your computer.



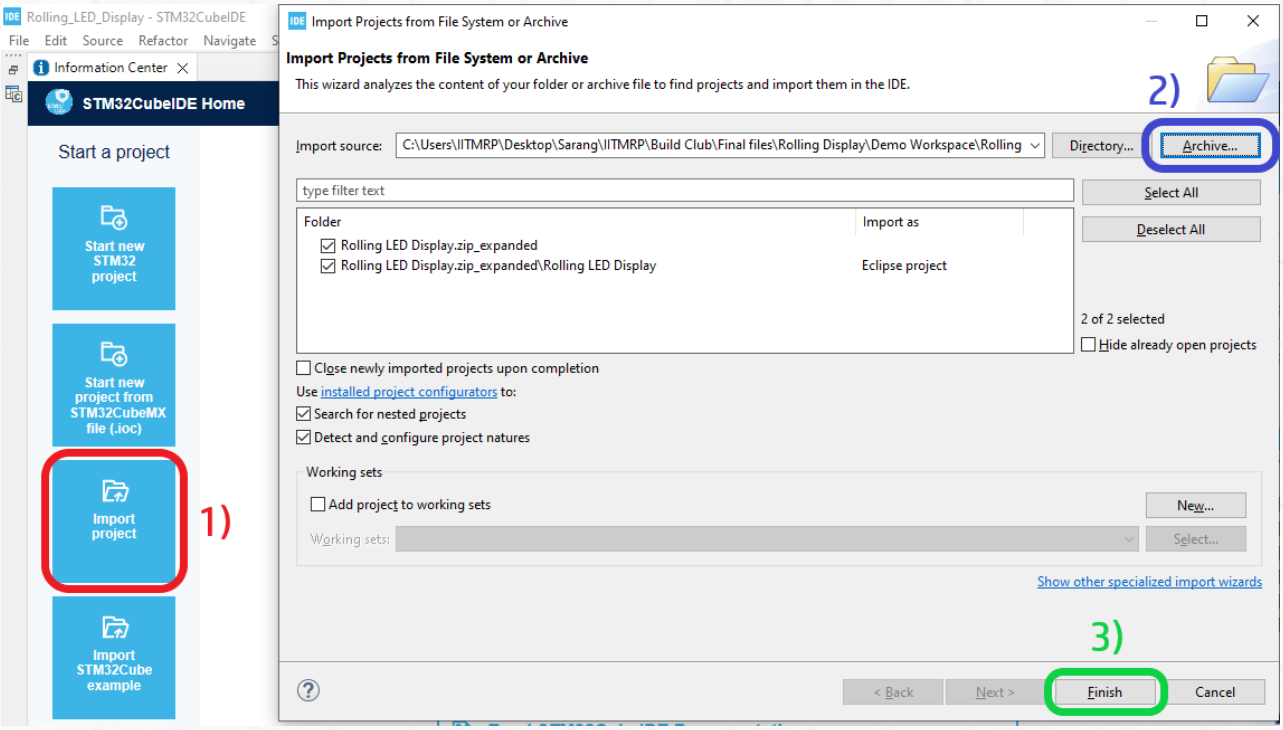
Step 2:

In the dialog box that opens, click 'Browse' and from the File Explorer window that opens, select the Rolling_LED_Display folder that you just created. Next, hit 'Launch'.



Step 3:

In the next window, choose 'Import project' from the list of options. In the dialog window that opens, Click 'Archive' and select the ZIP file ('Rolling LED Display.zip') from the location you downloaded it to. Click 'Finish'.

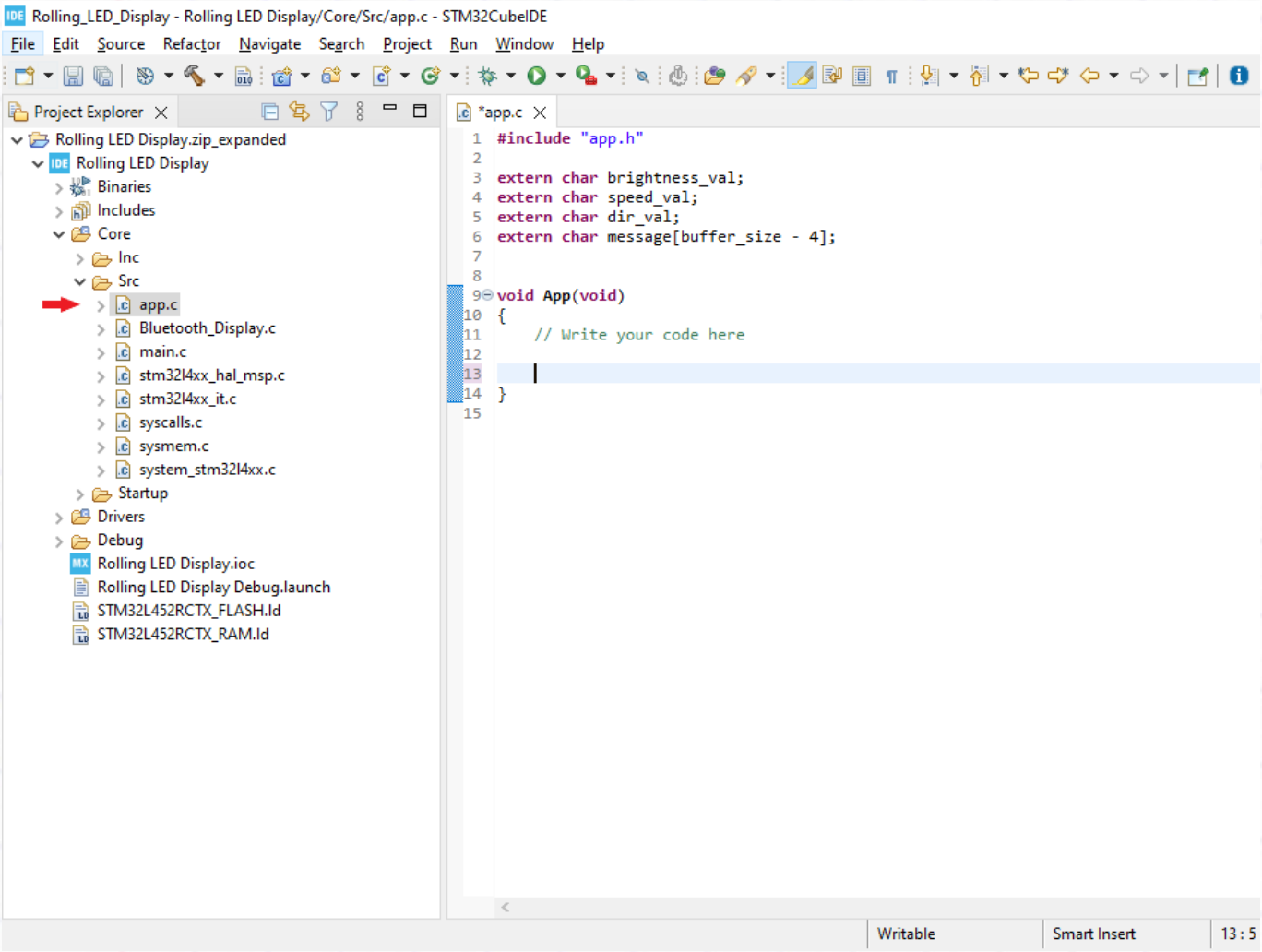


Step 4:

After importing the project successfully, you will see the below window. From the drop down on the left side, go to:

Rolling LED Display.zip_expanded > Rolling LED Display > Core > Src > app.c

Open app.c - this is where you will be writing your code



Code

(NOTE: All the code has to be written in the App() function in the app.c file.)

Level 1

In this level, we will learn to execute the project using minimal code.

Step 1: Scroll a message on the display

The Scroll_message function allows you to scroll a message with specified **direction**, **speed** and **brightness** on the display.

- Scroll_message (message, direction, speed, brightness);

  Input values: direction: 'L' or 'R', speed: 'H', 'M', 'L', brightness: 'H', 'M', 'L'

  eg. The following command scrolls the message "HELLO" in left direction at high speed and medium brightness, the following code is used:

  Scroll_message ("HELLO", 'L', 'H', 'M');

  Alternatively, variables containing each value can also be passed as inputs:

  Scroll_message (message, dir_val, speed_val, brightness_val);

Task:

Scroll a message from the IDE. Specify the direction, speed and brightness of the message in the code.

Step 2: Scroll a message continuously on the display

The Scroll_message function, when put inside an infinite **while** loop, allows you to scroll a message continuously on the display.

- while (1) // Whatever code is written inside it will be continuously executed in an infinite loop

```
while (1)                    // Whatever code is written inside it will be
{                               continuously executed in an infinite loop
        // Some code;
}
```

eg. To continuously scroll a message, the following code can be used:

```
while (1)
{
        Scroll_message ("HELLO", 'R', 'M', 'L');
}
```

Task:

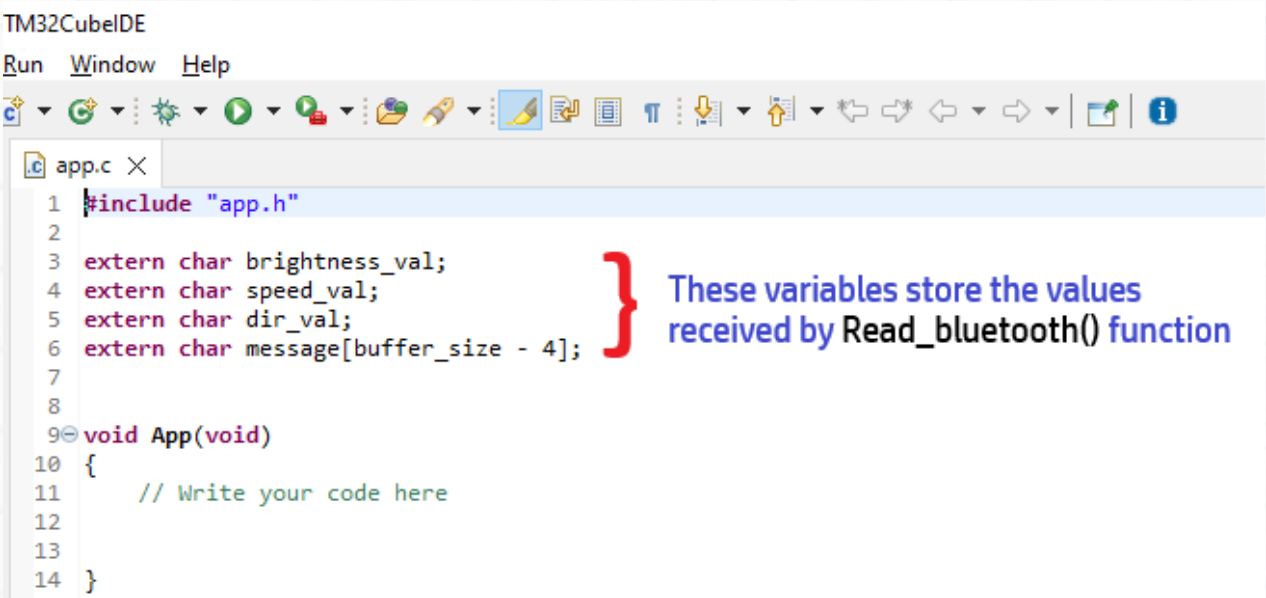
Make the message displayed in the previous task scroll continuously in an infinite loop.

Step 3: Scroll a message sent over Bluetooth continuously on the display.

The Read_bluetooth function is now used to read data sent from the Bluetooth terminal app on your phone. When combined with a small time **delay** and the Scroll_message function - all within a **while** loop, you can print messages on the display wirelessly from your phone.

- Read_bluetooth ();

Reads the bluetooth data sent from the phone. Stores the values of Direction, Speed, Brightness and Message sent by the user from the phone, inside the variables shown in the figure below.



To work properly, the **Read_bluetooth** function must always be followed with a small time delay. A Delay of 200ms is sufficient to get data over Bluetooth accurately.

- Delay_ms (time);

eg. To create a 200 ms time gap, write Delay_ms (200);

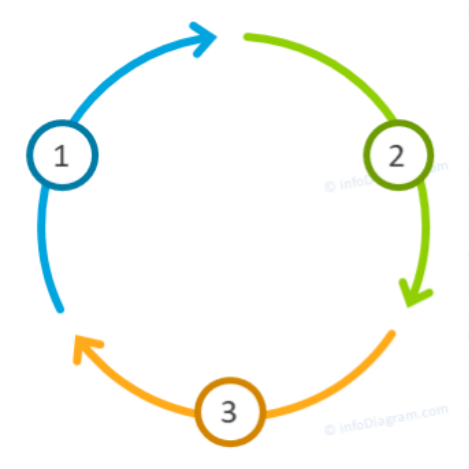
As the **Read_bluetooth** function needs to be continuously active in order to receive data at all times, it must only be used within a while loop. The variables storing the Bluetooth values mentioned before can then be passed as inputs to the Scroll_message function, which must also be within the same while loop.

Eg. Scroll_message (message, dir_val, speed_val, brightness_val);

NOTE: Remember to end each line of code with a semicolon ';'

Flow of the final code:

(1) Read Bluetooth
(2) Create a small time delay
(3) Scroll message on the display



Using the Bluetooth terminal app

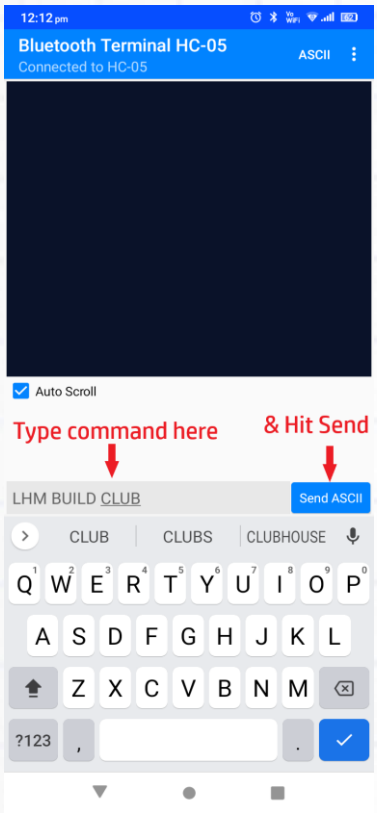
After pairing the HC-05 module to your phone using Bluetooth, open the HC-05 Terminal app and select 'HC-05' from 'Paired Devices'. You will now enter the communication window in the app as shown in the below image. The rapid blinking of the HC-05 module when disconnected will now switch to blinking with a frequency of 2s, once connected to the phone.

Bluetooth terminal syntax

XYZ MESSAGE

'X' : Direction ---> 'L' or 'R' - Left or Right
'Y' : Speed ---> 'H', 'M' or 'L' - High, Medium or Low
'Z' : Brightness ---> 'H', 'M' or 'L' - High, Medium or Low
'MESSAGE' : Message to be displayed

Note the example shown in the snapshot —>

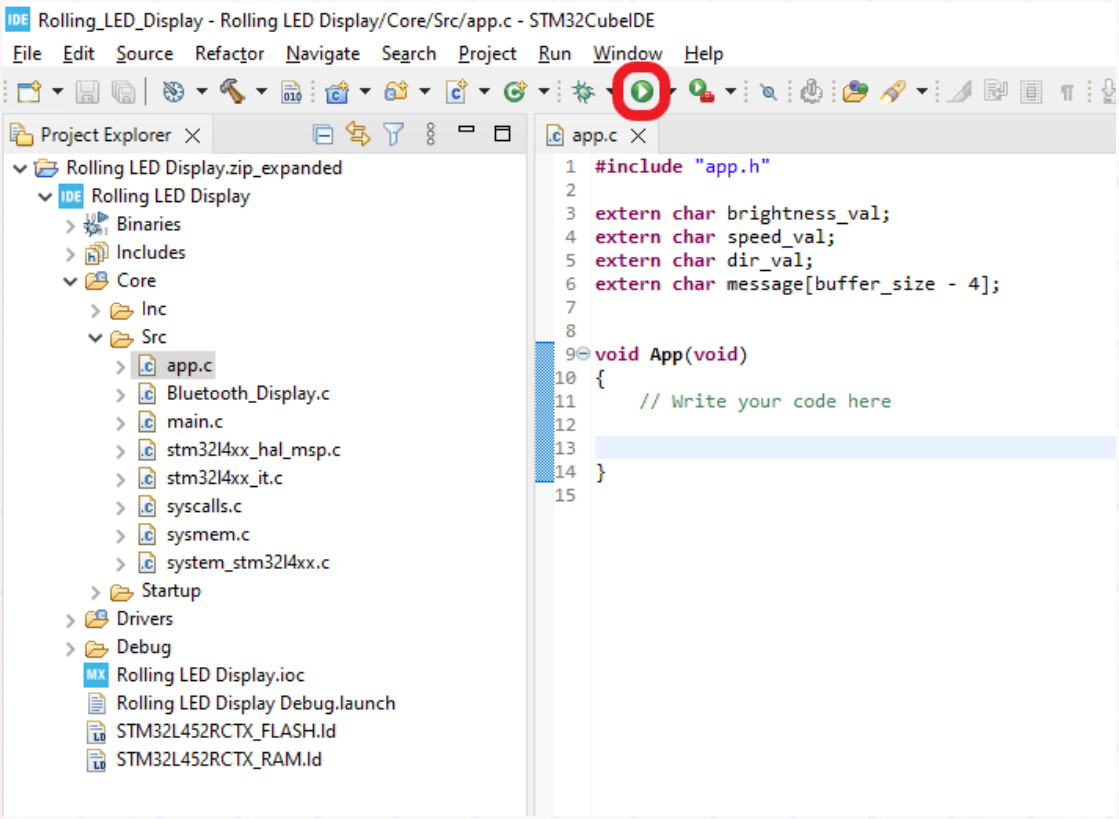


Task:

Write a program so that you can receive the message and values of direction, speed, brightness wirelessly from your phone, instead of directly from the IDE. This can be done using the HC-05 Terminal app installed earlier. Scroll the message with specified parameters continuously on the display.

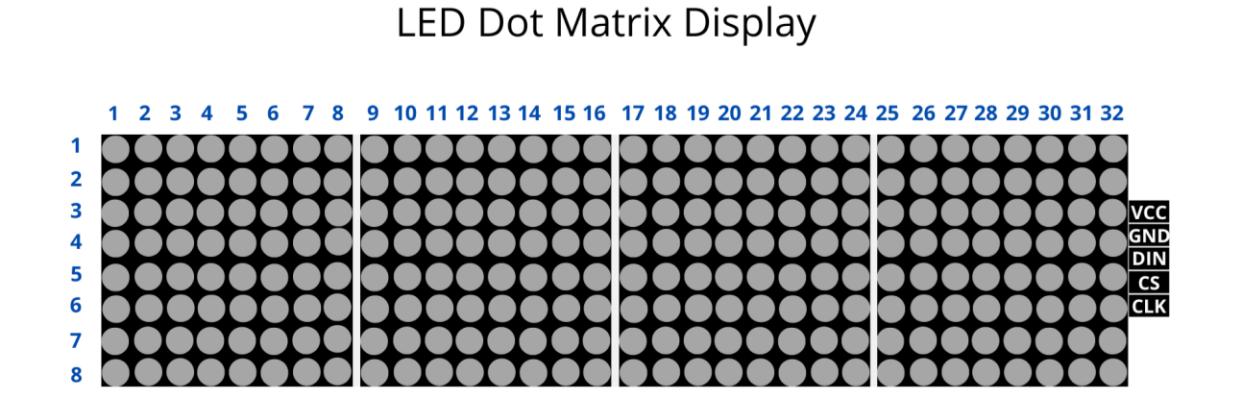
Uploading the Code to the Dexter

After writing the code in the specified space and building it, we can upload the code to the Dexter board by clicking the 'RUN' button in the IDE as shown below.



Level 2

In this level, we will learn in-depth how to control the LED Display step-by-step and will also create our own symbols / emojis to display. Finally, as a challenge, you will plan and run an animation on the display!



Step 1: Control specified LEDs of the display

The Set_LED function allows you to Turn ON / OFF any LED of the display by specifying its Row and Column numbers.

- Set_LED (row, column, state);

  row = row value of the LED ---> input values: 1 to 8
  column = column value of the LED ---> input values: 1 to 32
  state = 1 for ON or 0 for OFF

  Eg. To Turn ON LED at row 8, column 23:Set_LED(8, 23, 1);

Tasks:

1) Turn ON LED at 1st row, 3rd column (1, 3). Turn it OFF.
2) Turn ON LED at 8th row, 31st column (8, 31). Wait 500ms and turn it OFF.
3) Blink LED at 2nd row, 25th column (2, 25) for a duration of 1.5s with an interval of 1s, in an infinite (While) loop.
4) Turn ON LEDs at (1,4), (2,3), (2,5), (3,2), (3,6), (4,2), (4,6), (5,2), (5,3), (5,4), (5,5), (5,6), (6,2), (6,6), (7,2), (7,6). What pattern do you see?
5) Create a pattern where an LED light at (2, 1) shifts rightward until it reaches (2, 32).
6) Repeat task 5, but now when the LED light reaches (2, 32), it should re-enter from (2,1) and start over

Step 2: Program a column of LEDs at once

The Set_col function allows you to set all the LEDs of a column at one go.

- Set_col (column, pattern);

  column = column number selected ---> input values: 1 to 32
  pattern = binary / decimal / hexadecimal pattern corresponding to LEDs, where 1's represent ON LEDs and 0's represent OFF LEDs.

  Eg. To Turn ON alternate LEDs of column 2: Set_col (2, 0b10101010);

  NOTE: 0b10101010 (binary) can also be written as 0xAA in hexadecimal or 170 in decimal, hence Set_col (2, 0xAA); or Set_col (2, 170); can also be used. (Refer the Prerequisite video on Number Systems.)

Tasks:

1) Turn ON 2nd, 3rd, 5th & 6th LEDs of column 2 using Set_col function.

2) Make the above pattern scroll from column 2 to column 23.

3) Make the above pattern scroll from column 1 to column 32 and re-enter from column 1 and start over - in an infinite loop.

4) Write the following values using Set_col function:

254 to column 26	60 to column 9
64 to column 27	66 to column 10
32 to column 28	169 to column 11
32 to column 29	133 to column 12
64 to column 30	133 to column 13
254 to column 31	169 to column 14
	66 to column 15
What pattern do you see?	60 to column 16

Step 3: Write letters to the display & control their brightness

The Write_letter function allows you to write any uppercase letter ('A' to 'Z') by passing it as an input. You can select the 8x8 display to which you want to write. The **Set_brightness** function allows you to control the display's brightness.

- Write_letter (letter, display number);

  letter = uppercase letter to be printed ---> input values: 'A' to 'Z'
  display number = 8x8 display number to be written to ----> input values: 1-4 (since you have four 8x8 displays)

Eg. To write 'K' to 3rd display: Write_letter ('K', 3);

- Set_brightness (brightness);

  brightness = brightness of display ---> input values: 0-15 (0 is min brightness, 15 is max brightness)

Eg. To set medium brightness: Set_brightness (5);

Tasks:

1) Write 'A' to 1st display.
2) Write "IITM" on the display by writing each letter individually to a display.
3) Set brightness to 0 and write something on the display. Wait 2 seconds and set brightness to 15.
4) Write "IITM" on the display as in task 2. Now, write a code so that the brightness gradually increases from 0 to 15 and then gradually fades from 15 to 0 (use While and For loops for this - taught in C prerequisite).

Step 4: Shift the display data Leftward & Rightward

- Shift_left ();

  Shifts the display data towards the Left.

- Shift_right ();

  Shifts the display data towards the Right.

- Shift_letter (letter, direction, speed);

  letter = Letter to be written ---> input values: 'A' to 'Z' & blank space (' ')
  direction = direction of scroll ---> input values: 'L' for left, 'R' for right
  speed = scrolling speed ---> input values: 0-10 (10 is max speed)

Eg. To shift letter 'M' towards left direction at medium speed: Shift_letter('M','L', 5);

Tasks:

1) Write "IITM" on the display. Shift the data Left 2 times, followed by Right 4 times and again Left 2 times.

2) Write "IITM" on the display. This time shift the data Left 4 times, followed by Right 4 times. Did you notice anything? Why do you think this happened and how can it be resolved?

3) Use the Shift_letter function to shift the letter 'V' at the highest possible speed in the Left direction.

4) Now shift the following letters one after another: 'W', 'E', 'L', 'C', 'O', 'M', 'E'. Try it for Left direction & Right direction? What insight did you get from this? Find a solution to the glitch you observed and write code for it.

5) Above the App() function in app.c, create a String variable named 'word' as follows and write any word between the 2 quotation marks:

app.c

```c
#include "app.h"

char word[] = "BUILD";

void App(void)
{
    // Write your code here


}

```

Now write code inside the App() function to scroll the 'word' variable on the display. You will require the Shift_letter function, a For loop and will need to know string indexing for this (can be learnt from the C resources given in the 'Prerequisites' section of this document).

6) Continuing from task 5, using an infinite While loop, scroll the same word continuously so that once the word scrolls and completely leaves the display from one end, it re-enters the display from the other end and starts over. For this, the Shift_left and Shift_right functions will also be required additionally.

You now know how to control the LED Dot Matrix Display!

Challenges:

1) Write code to print digits 0-9 on the display. Try similarly for punctuation marks '!', '?', ';'.
2) Design 3 emojis or symbols of your choice and print them on the display.
3) Write a program that animates a 'Stick man' walking from one end of the display to the other.
4) Plan an animation clip of minimum 10 seconds with a storyline and run it on the display. Whoa! You just built your own TV! Take a video clip of the animation movie and post it on the Build Club Discord Server.