

g^o ganeshprabhu2005 /

EXPERIMENT--05-SOIL-MOISTURE-SENSOR-INTERFACE-TO-IOT-DEVELOPMENT-BOARD-

Public

forked from [vasanthkumarch/EXPERIMENT--05-SOIL-MOISTURE-SENSOR-INTERFACE-TO-IOT-DEVELOPMENT-BOARD-](#)

☆ 0 stars g^o 122 forks g^o Branches Tag Tags Activity

Star

Notifications

<> Code Pull requests Actions Projects Security Insights



This branch is [2 commits ahead](#) of

[vasanthkumarch/EXPERIMENT--05-SOIL-MOISTURE-SENSOR-INTERFACE-TO-IOT-DEVELOPMENT-BOARD- :main](#).



ganeshprabhu2005 Update README.md

now



README.md

Update README.md

now



README



EXPERIMENT--05-SOIL-MOISTURE-SENSOR-INTERFACE-TO-IOT-DEVELOPMENT-BOARD-

Aim: To Interface a Analog Input (soil moisture sensor) to ARM IOT development board and write a program to obtain the data on the com port

Components required: STM32 CUBE IDE, ARM IOT development board, STM programmer tool.

Theory

Hardware Overview

A typical soil moisture sensor consists of two parts.

The Probe The sensor includes a fork-shaped probe with two exposed conductors that is inserted into the soil or wherever the moisture content is to be measured.

As previously stated, it acts as a variable resistor, with resistance varying according to soil moisture.

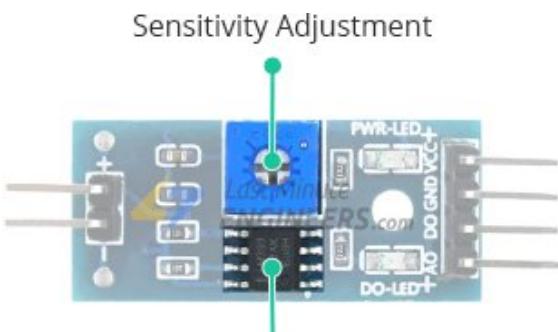


The Module In addition, the sensor

includes an electronic module that connects the probe to the Arduino.

The module generates an output voltage based on the resistance of the probe, which is available at an Analog Output (AO) pin.

The same signal is fed to an LM393 High Precision Comparator, which digitizes it and makes it available at a Digital Output (DO) pin.



LM393 Comparator

The module includes a potentiometer for adjusting the sensitivity of the digital output (DO).

You can use it to set a threshold, so that when the soil moisture level exceeds the threshold, the module outputs LOW otherwise HIGH.

This setup is very useful for triggering an action when a certain threshold is reached. For example, if the moisture level in the soil exceeds a certain threshold, you can activate a relay to start watering the plant.

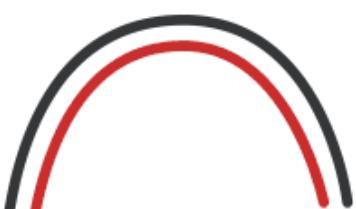
Soil Moisture Sensor Pinout The soil moisture sensor is extremely simple to use and only requires four pins to connect.

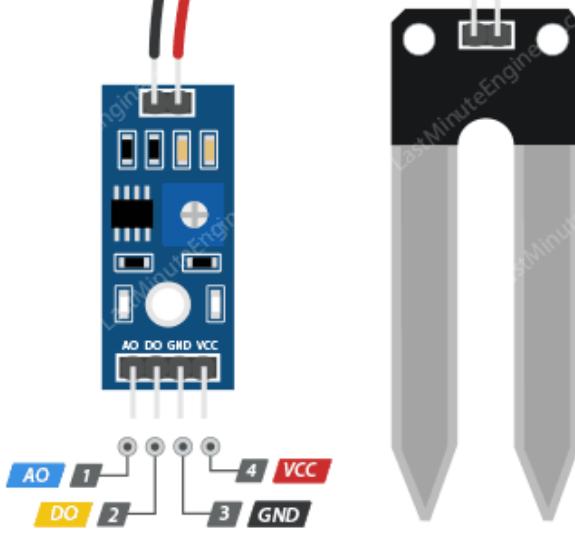
soil moisture sensor pinout AO (Analog Output) generates analog output voltage proportional to the soil moisture level, so a higher level results in a higher voltage and a lower level results in a lower voltage.

DO (Digital Output) indicates whether the soil moisture level is within the limit. D0 becomes LOW when the moisture level exceeds the threshold value (as set by the potentiometer), and HIGH otherwise.

VCC supplies power to the sensor. It is recommended that the sensor be powered from 3.3V to 5V. Please keep in mind that the analog output will vary depending on the voltage supplied to the sensor.

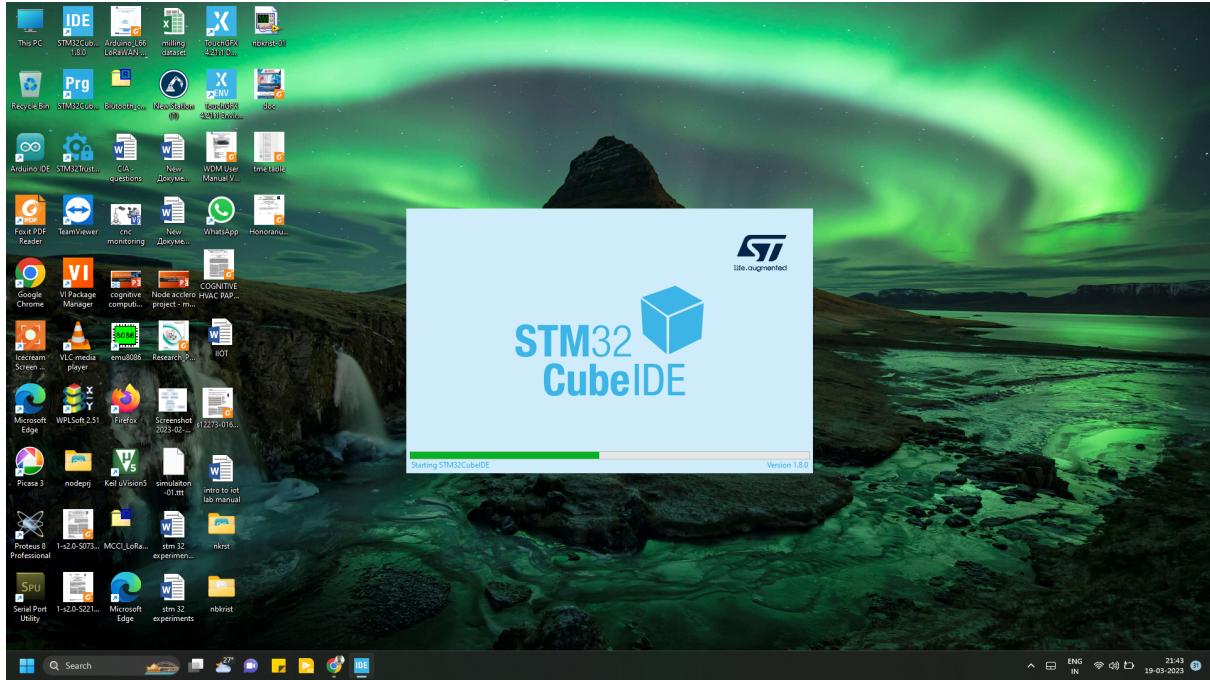
GND is the ground pin.



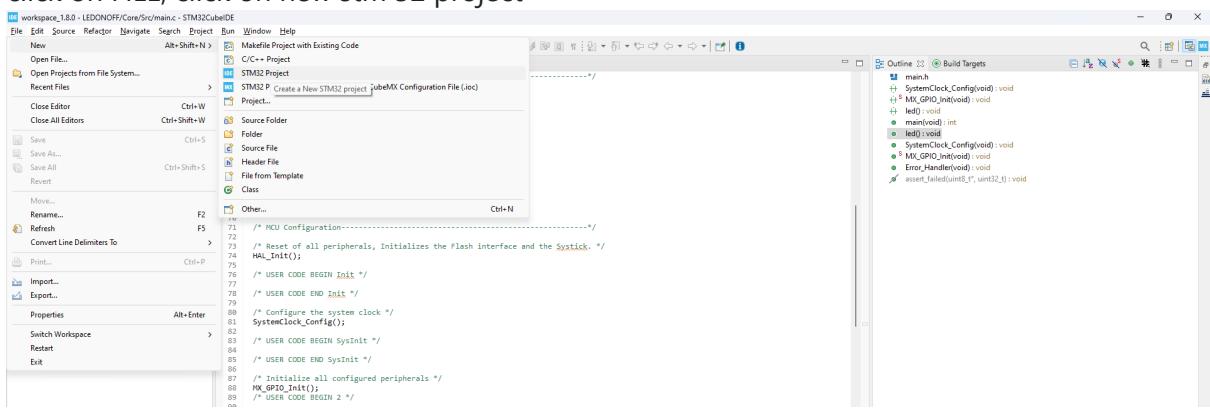


Procedure:

1. click on STM 32 CUBE IDE, the following screen will appear



2. click on FILE, click on new stm 32 project



```

91  /* USER CODE END 2 */
93  /* Infinite loop */
94  /* USER CODE BEGIN WHILE */
95  while (1)
96  {
97      led();
98      /* USER CODE END WHILE */
99  }
100 /* USER CODE BEGIN 3 */
101 /* USER CODE END 3 */
102 }
103 }

104 void led()
105 {
106     HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
107     HAL_Delay(3000);
108 }

0 items selected

```

workspace_1.8.0 - LEONOR/Core/main - STM32CubeIDE

File Edit Source Refactor Search Project Run Window Help

Project Explorer

- > main.c
- > adc
- > DEMO
- > exp-01
- > EXP-02
- > EXP-03
- > interrupt
- > interruptconcept
- > IR_Sensor_Interfacing_Digital_Read (in IR_GPIO_READ)
- > LoRaWAN_End_Nodes (in STM32CubeIDE)
- > LQBOARD_USART (in LQBOARD)
- > lux_sensor
- > proj
- > PUSHBUTTON
- > Soil moisture sensor
- > ssrss

Outline Build Targets

Progress Information

Initializing STM32 Target Selector...

Cancel

No consoles to display at this time.

3. select the target to be programmed as shown below and click on next

STM32 Project

Target Selection

Select STM32 target or STM32Cube example

MCU/MPU Selector | Board Selector | Example Selector | Cross Selector

Part Number: STM32G071RB

Core: Arm Cortex-A7 + Arm Cortex-M4

Series: STM32F0

Features: STM32G071RB

Mainstream Arm Cortex-M0+ MCU with 128 Kbytes of Flash memory memory, 36 Kbytes RAM, 64 MHz CPU, 4x USART, timers, ADC, DAC, comm. If, 1.7-3.6V

Product is in mass production

Unit Price for 10kU (USD): 1.803

Boards: NUCLEO-G071RB - STM32G071B-DISCO

LQFP64

The STM32G071xB mainstream microcontrollers are based on high-performance Arm® Cortex®-M0+ 32-bit RISC core operating at up to 64 MHz frequency. Offering a high level of integration, they are suitable for a wide range of applications in consumer, industrial and appliance domains and ready for the Internet of Things (IoT) solutions. The devices incorporate a memory protection unit (MPU), high-speed embedded memories (up to 128 Kbytes of Flash program memory with read protection, write protection, proprietary code protection, and securable area, and 36 Kbytes of SRAM), DMA and an extensive range of system functions, enhanced I/Os and peripherals. The devices offer standard communication interfaces (two I²Cs, two SPIs / one I²S, one HDMI CEC, and four USARTs), one 12-bit ADC (2.5 MSPS), up to 19 timers, two 12-bit DACs with two channels, two fast comparators, an internal voltage reference buffer, a low-power RTC, an advanced control PWM timer running at up to double the CPU frequency, general purpose timers with one timer running at double the CPU frequency, a 32-bit general-purpose timer, two basic timers, two low-power 16-bit timers, two watchdog timers, and a SysTick timer. The STM32G071xB devices provide a fully integrated USB Type-C Power Delivery controller.

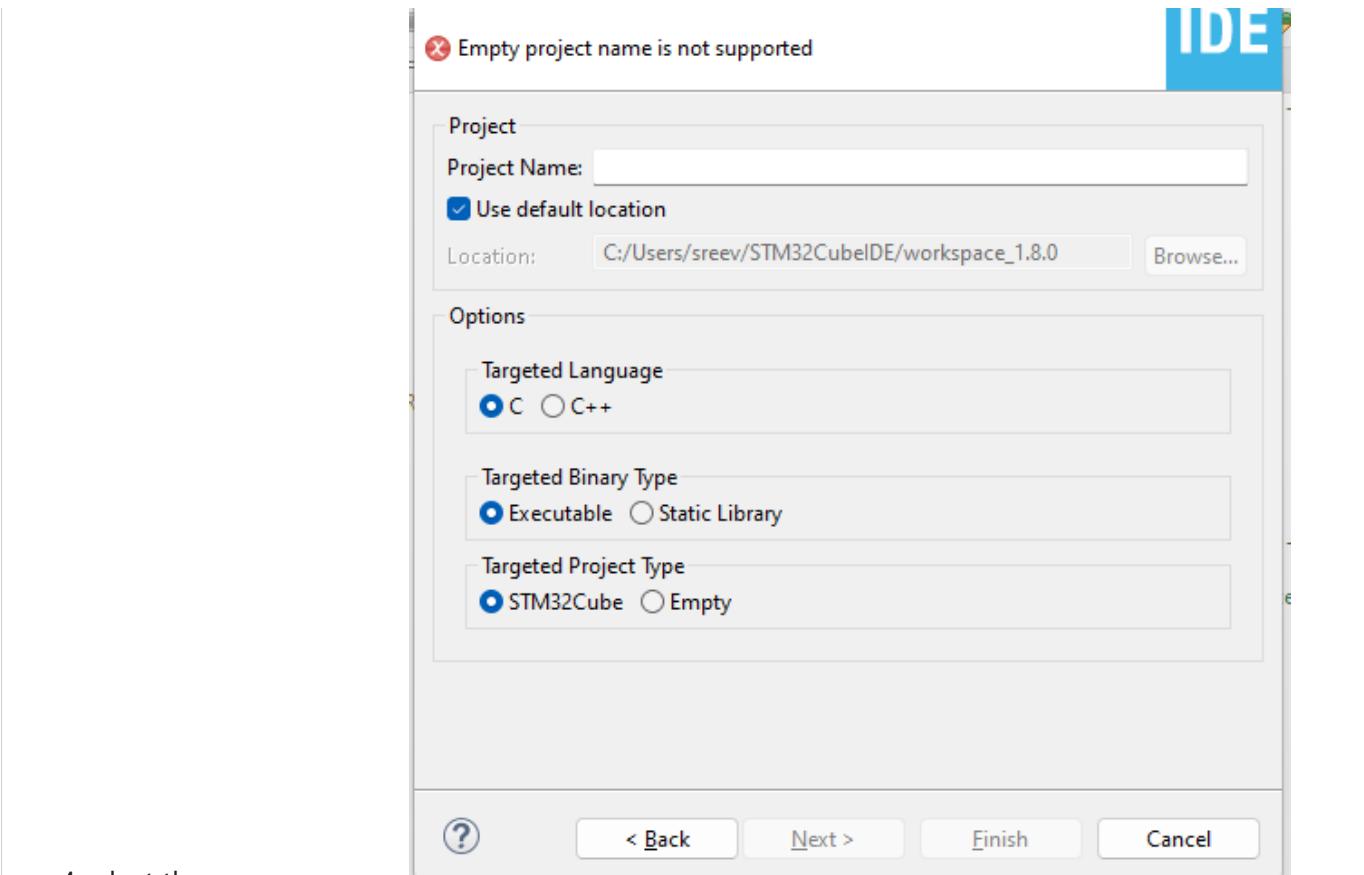
The devices operate within ambient temperatures from -40 to 125°C. They can operate with supply voltages from 1.7 V to 3.6 V. Optimized dynamic consumption combined with a comprehensive set of power-saving modes, low-power timers and low-power UART, allows the design of low-power applications.

MCUs/MPUs List: 2 items

	Part No	Reference	Marketing Status	Unit Price for 10kU (USD)	Board	Package	Flash	RAM	I/O	Freq
<input checked="" type="checkbox"/>	STM32G071...	STM32G071...	Active	1.803	NUCLEO-G0...	UFBGA64	128 kBytes	36 kBytes	60	64 MHz
<input checked="" type="checkbox"/>	STM32G071...	STM32G071...	Active	1.803	NUCLEO-G0...	LQFP64	128 kBytes	36 kBytes	60	64 MHz

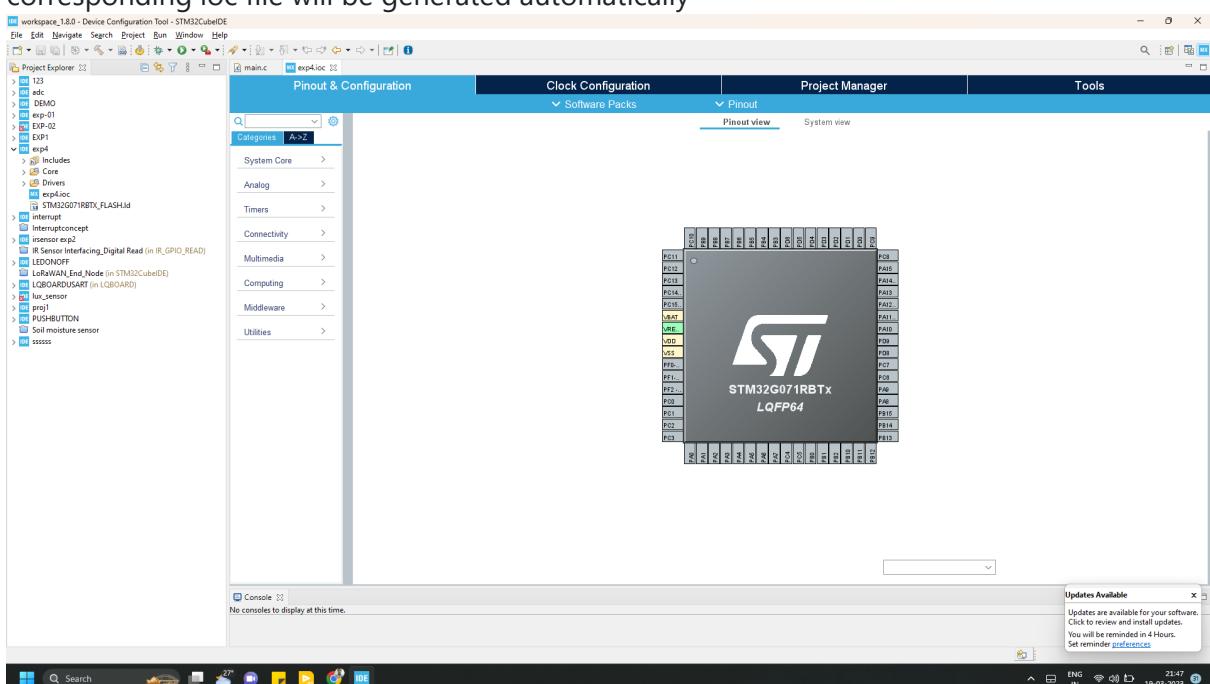
Next > | Back | Finish | Cancel

IDE STM32 Project

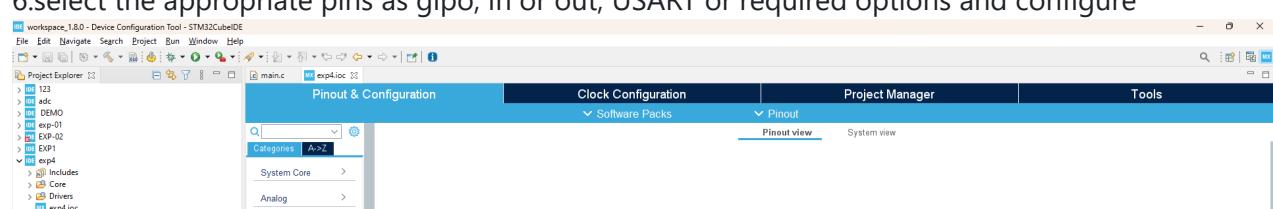


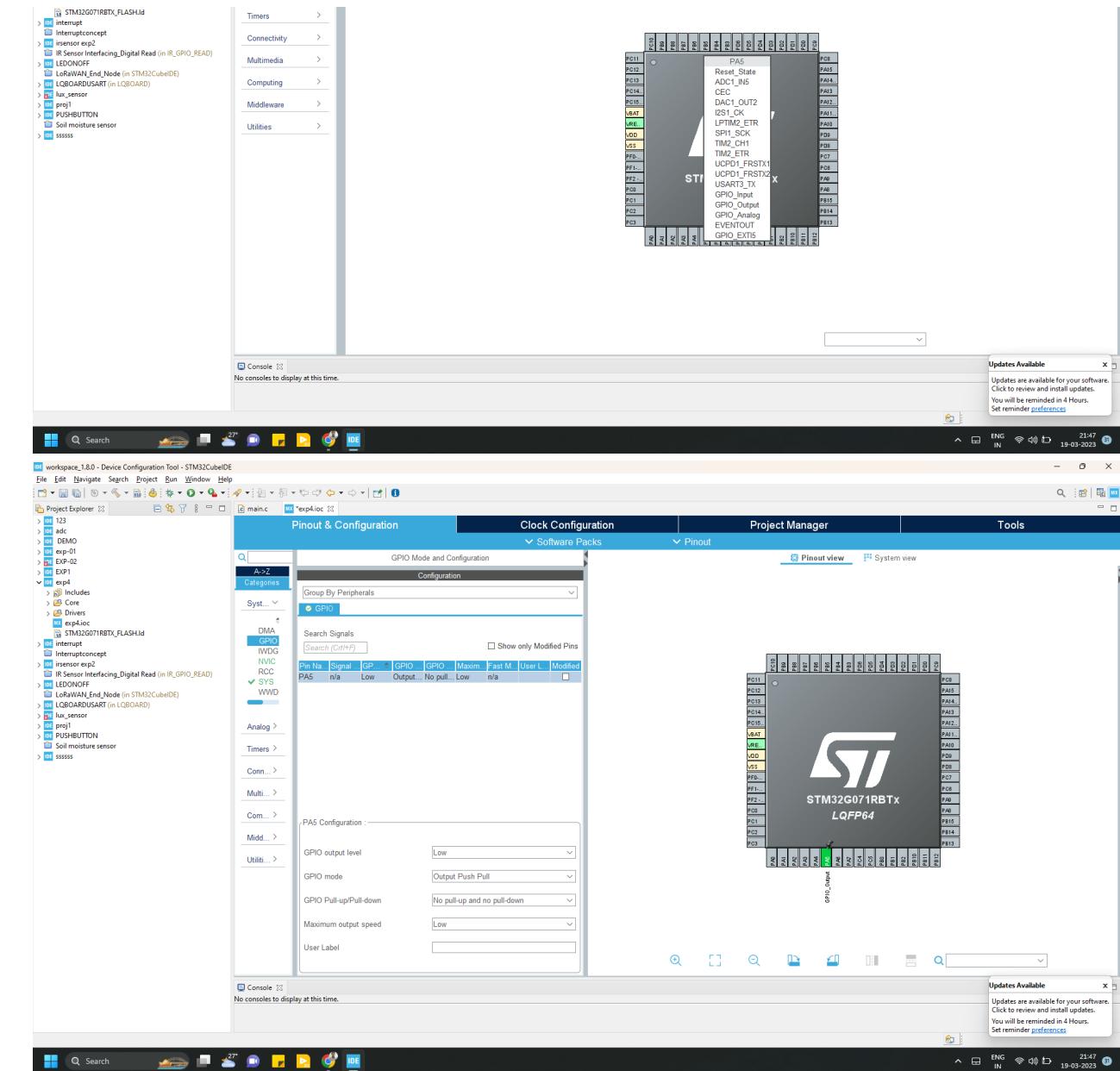
4.select the program name

5. corresponding ioc file will be generated automatically

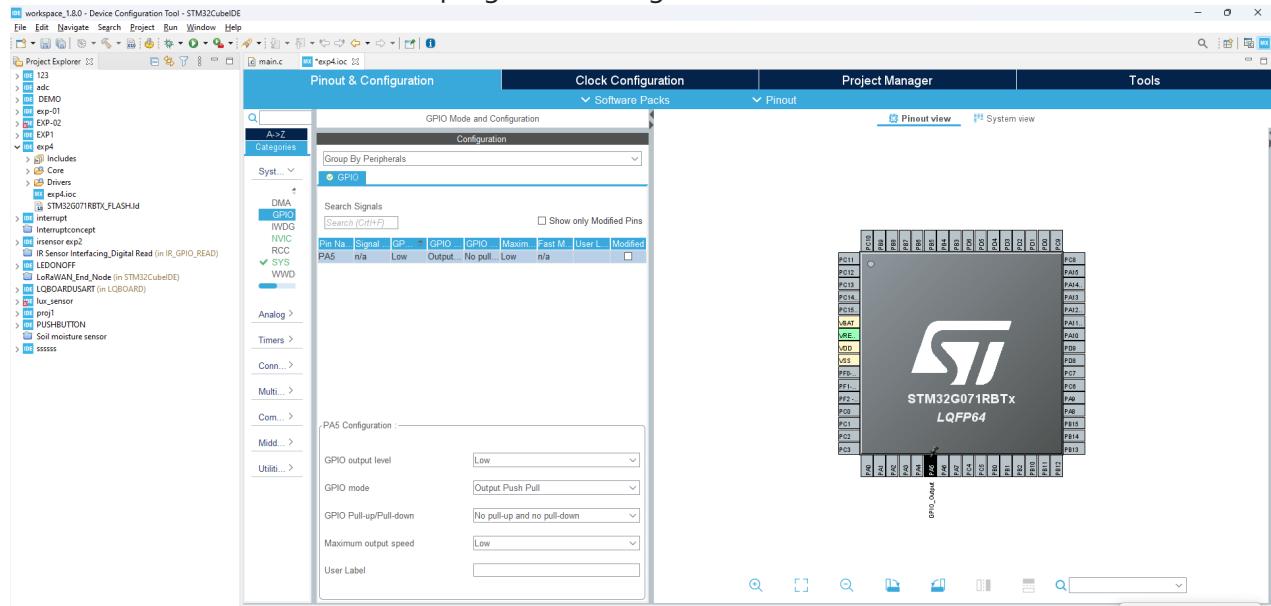


6.select the appropriate pins as gipo, in or out, USART or required options and configure





7.click on cntrl+S , automaticall C program will be generated



The screenshot shows the STM32CubeIDE interface with the main.c file open in the code editor. A progress dialog box titled "Device Configuration Tool Updating Code..." is displayed in the center. The code in main.c includes comments for user code sections and defines, such as:

```

1 /* USER CODE BEGIN Header */
2
3 /**
4  * @file       : main.c
5  * @brief      : Main program body
6  */
7
8
9 // Copyright (c) 2023 STMicroelectronics.
10 // All rights reserved.
11
12 // This software is licensed under terms that can be found in the LICENSE file
13 // in the root directory of this software component.
14 // If no LICENSE file comes with this software, it is provided AS-IS.
15
16
17 /* USER CODE END Header */
18
19 /* Includes */
20 #include "main.h"
21
22 /* Private includes */
23 /* USER CODE BEGIN Includes */
24
25 /* USER CODE END Includes */
26
27 /* Private typedef */
28 /* USER CODE BEGIN PTD */
29
30 /* USER CODE END PTD */
31
32 /* Private define */
33 /* USER CODE BEGIN PD */
34 /* USER CODE END PD */
35
36 /* Private macro */
37 /* USER CODE BEGIN PM */
38
39 /* USER CODE END PM */
40
41 /* Private variables */
42
43 /* USER CODE BEGIN PV */
44
45 /* USER CODE END PV */
46
47 /* Private function prototypes */
48 void SystemClock_Config(void);
49 /* USER CODE BEGIN FPP */
50
51 /* USER CODE END FPP */
52
53 /* Private user code */
54

```

8. edit the program and as per required

The screenshot shows the STM32CubeIDE interface with the main.c file open in the code editor. A progress dialog box titled "Device Configuration Tool Updating Code..." is displayed in the center. The code in main.c includes comments for user code sections and defines, such as:

```

1 /* USER CODE BEGIN Header */
2
3 /**
4  * @file       : main.c
5  * @brief      : Main program body
6  */
7
8
9 // Copyright (c) 2023 STMicroelectronics.
10 // All rights reserved.
11
12 // This software is licensed under terms that can be found in the LICENSE file
13 // in the root directory of this software component.
14 // If no LICENSE file comes with this software, it is provided AS-IS.
15
16
17 /* USER CODE END Header */
18
19 /* Includes */
20 #include "main.h"
21
22 /* Private includes */
23 /* USER CODE BEGIN Includes */
24
25 /* USER CODE END Includes */
26
27 /* Private typedef */
28 /* USER CODE BEGIN PTD */
29
30 /* USER CODE END PTD */
31
32 /* Private define */
33 /* USER CODE BEGIN PD */
34 /* USER CODE END PD */
35
36 /* Private macro */
37 /* USER CODE BEGIN PM */
38
39 /* USER CODE END PM */
40
41 /* Private variables */
42
43 /* USER CODE BEGIN PV */
44
45 /* USER CODE END PV */
46
47 /* Private function prototypes */
48 void SystemClock_Config(void);
49 static void MX_GPIO_Init(void);
50 /* USER CODE BEGIN FPP */
51
52 /* USER CODE END FPP */
53

```

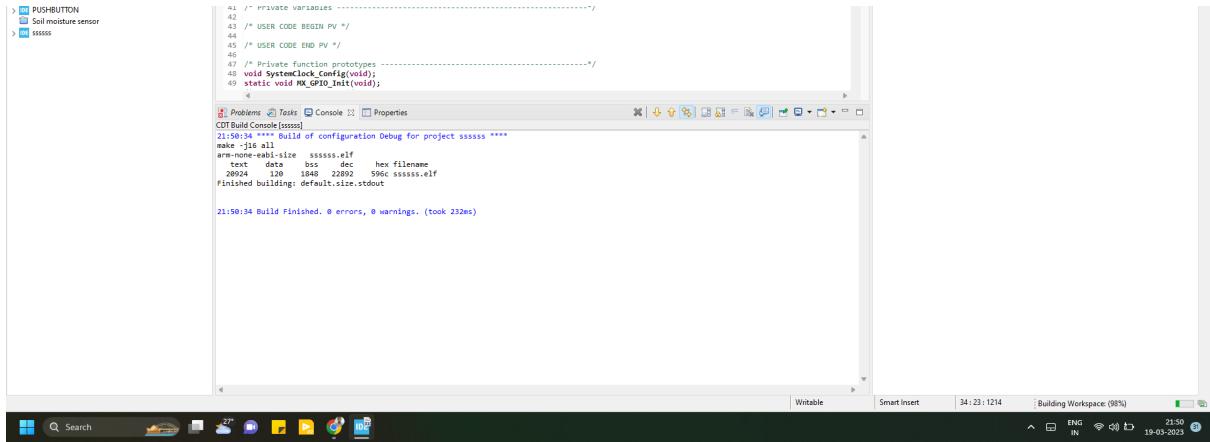
9. use project and build all

The screenshot shows the STM32CubeIDE interface with the main.c file open in the code editor. A progress dialog box titled "Device Configuration Tool Updating Code..." is displayed in the center. The code in main.c includes comments for user code sections and defines, such as:

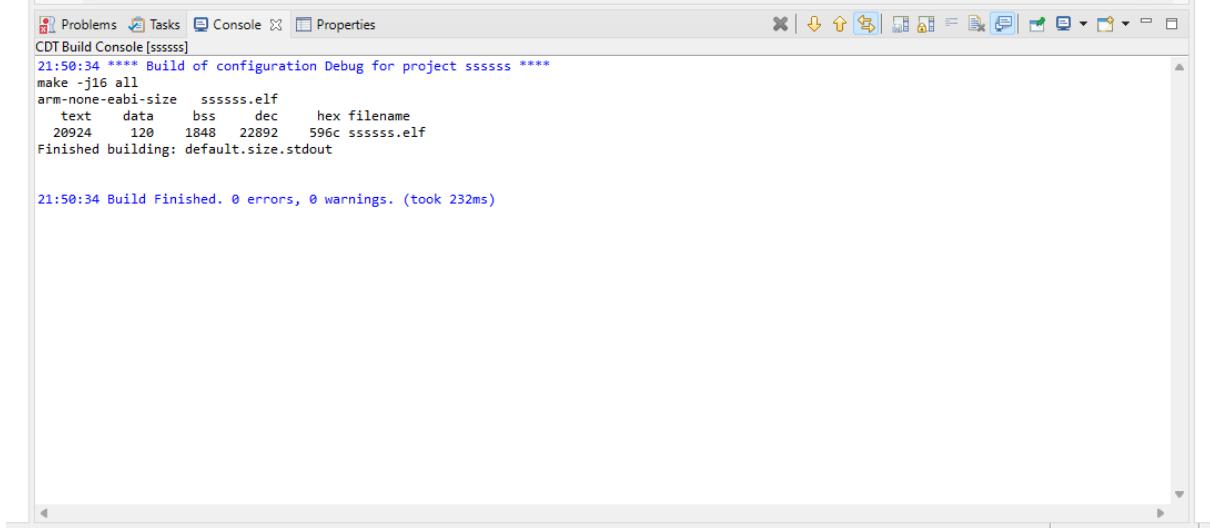
```

1 /* USER CODE BEGIN Header */
2
3 /**
4  * @file       : main.c
5  * @brief      : Main program body
6  */
7
8
9 // Copyright (c) 2023 STMicroelectronics.
10 // All rights reserved.
11
12 // This software is licensed under terms that can be found in the LICENSE file
13 // in the root directory of this software component.
14 // If no LICENSE file comes with this software, it is provided AS-IS.
15
16
17 /* USER CODE END Header */
18
19 /* Includes */
20 #include "main.h"
21
22 /* Private includes */
23 /* USER CODE BEGIN Includes */
24
25 /* USER CODE END Includes */
26
27 /* Private typedef */
28 /* USER CODE BEGIN PTD */
29
30 /* USER CODE END PTD */
31
32 /* Private define */
33 /* USER CODE BEGIN PD */
34 /* USER CODE END PD */
35
36 /* Private macro */
37 /* USER CODE BEGIN PM */
38
39 /* USER CODE END PM */
40

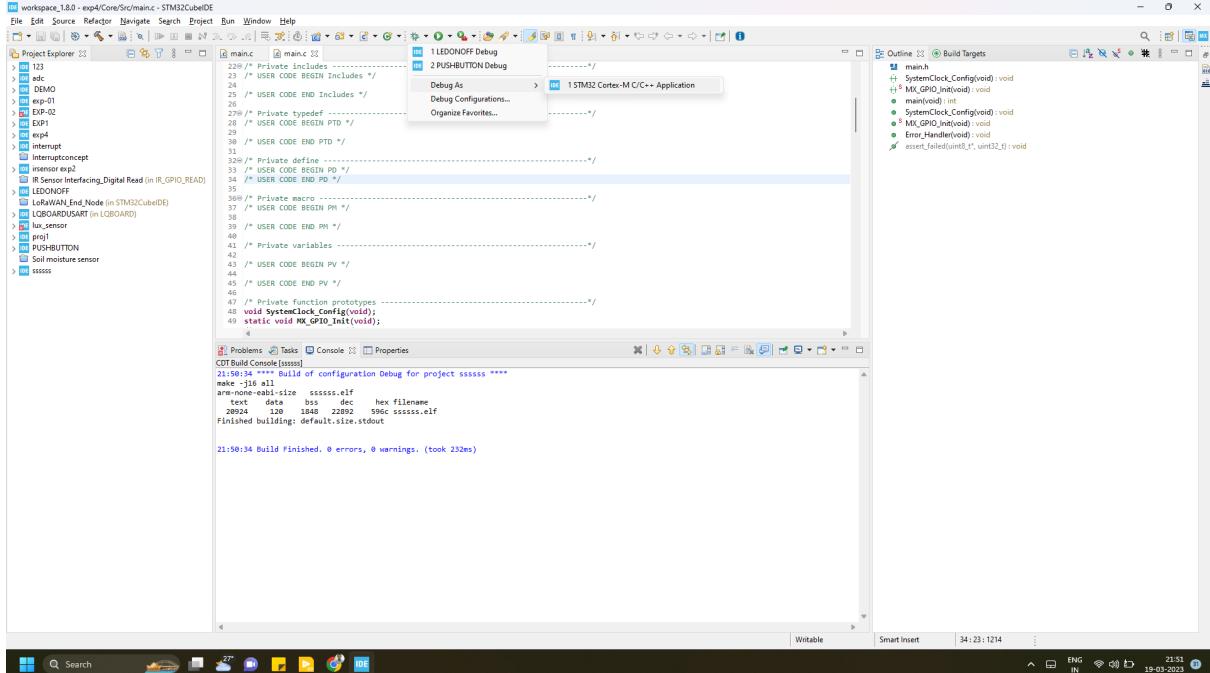
```



10. once the project is bulild

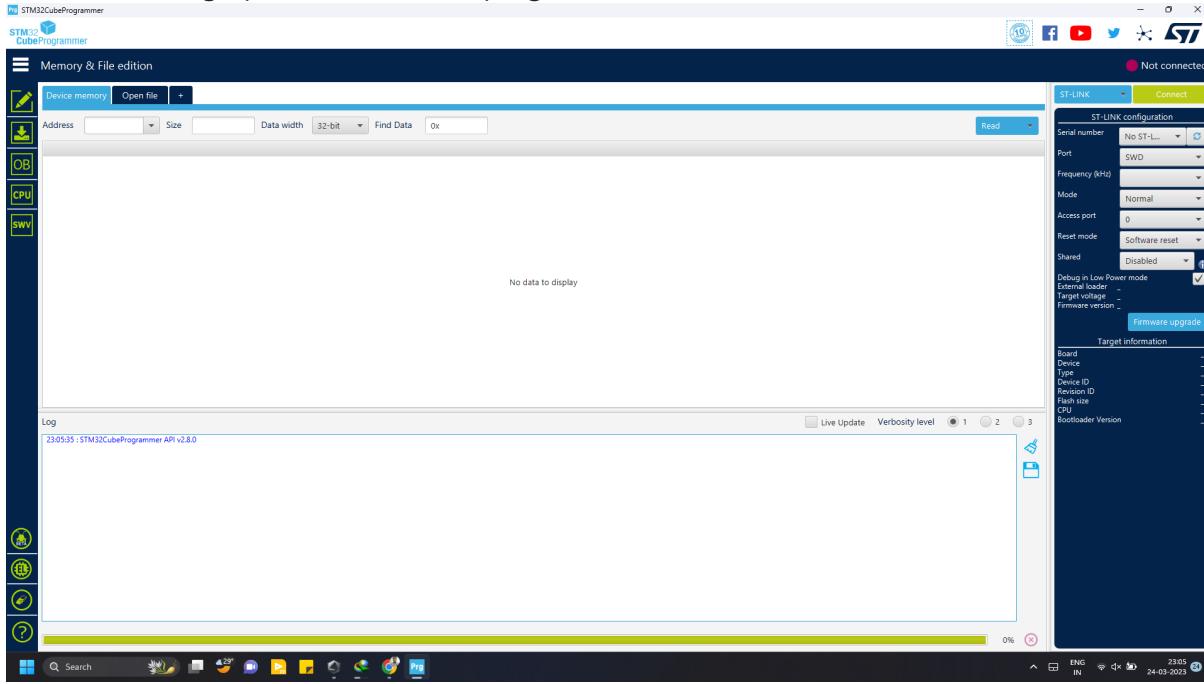


11. click on debug option

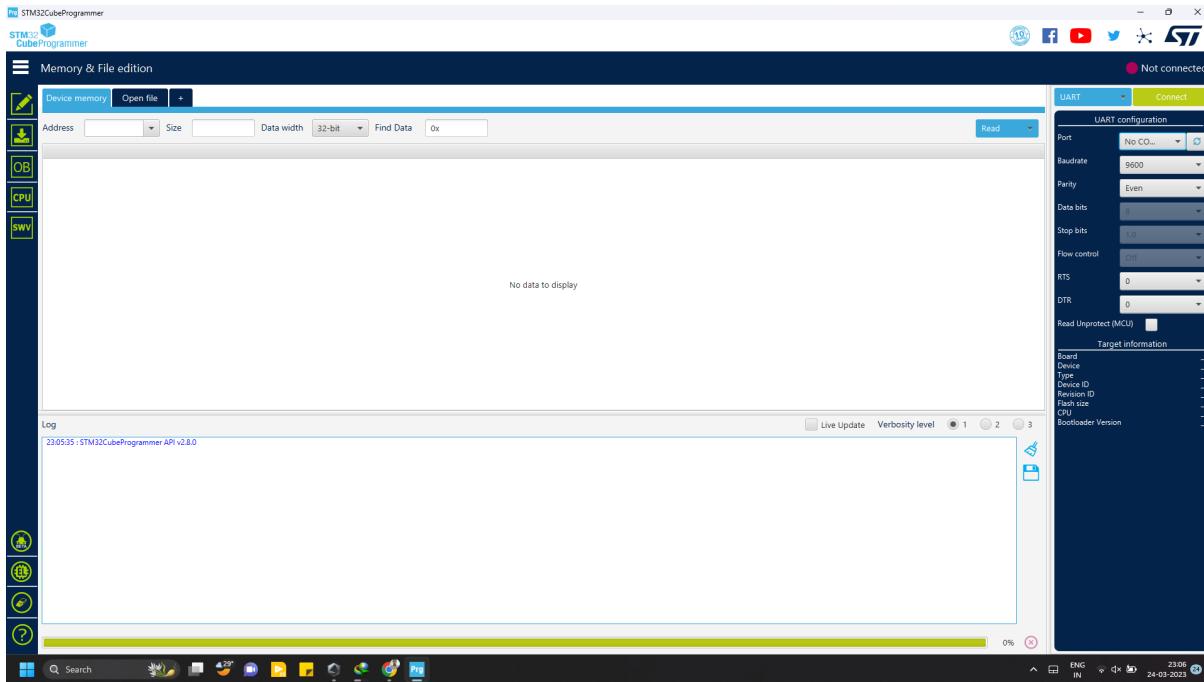


12. connect the iot board to power supply and usb

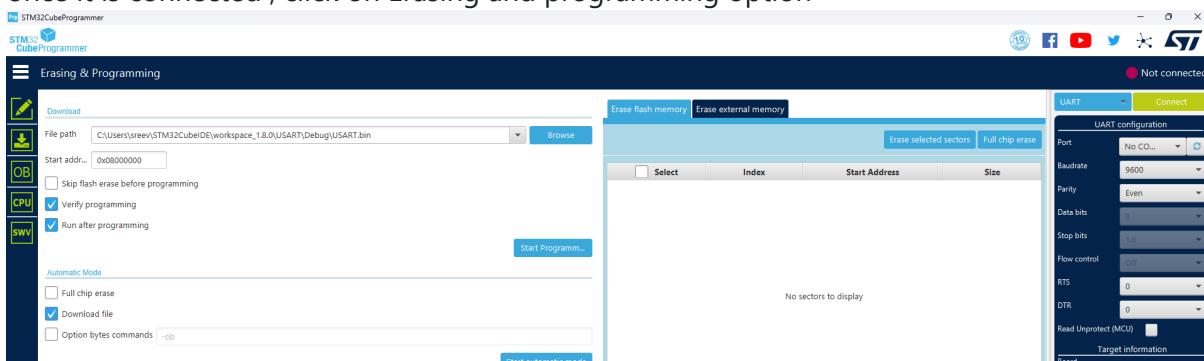
13. After connecting open the STM cube programmer

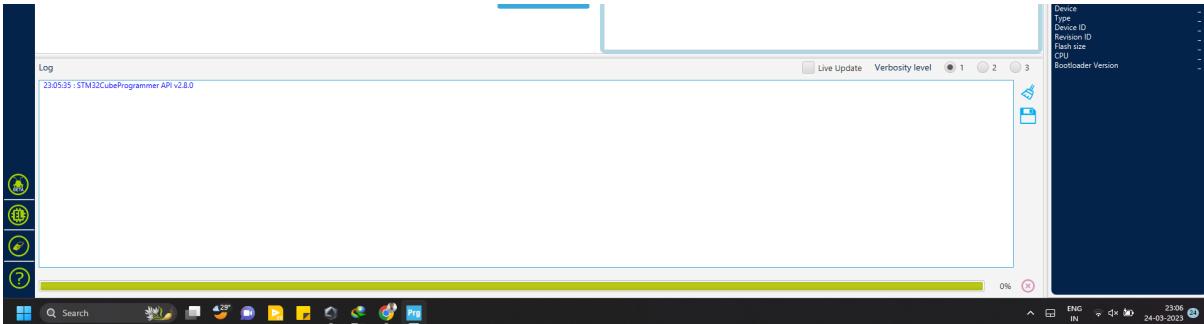


14. click on UART and click on connect

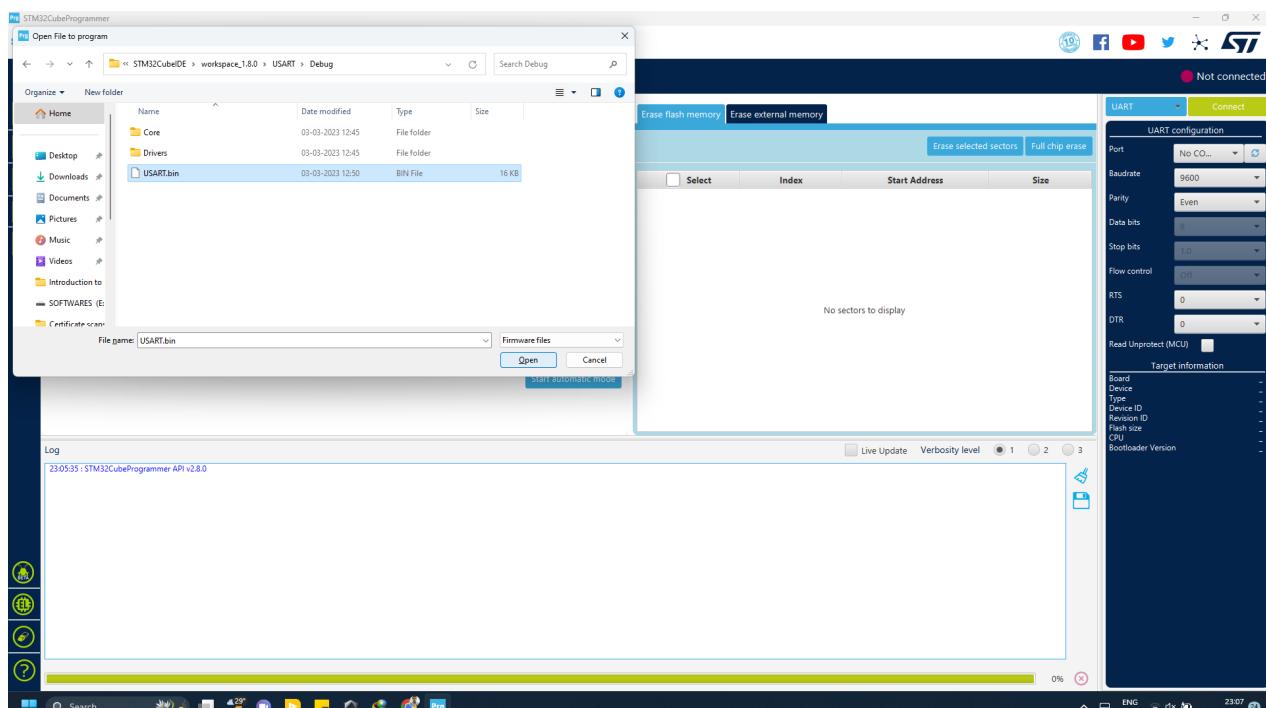


15. once it is connected , click on Erasing and programming option





16. flash the bin or hex file as shown below by switching the switch to flash mode



17. check for execution of the output by switching the board to run mode

18. open serial port utility and select appropriate com port, run and verify the results of moisture content .

STM 32 CUBE PROGRAM :

```
#include "main.h"
#include "stdio.h"

#if defined (_ICCARM) || defined (_ARMCC_VERSION)
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#elif defined(_GNUC_)
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#endif

PUTCHAR_PROTOTYPE
{
    HAL_UART_Transmit(&huart2, (uint8_t *)&ch, 1, 0xFFFF);
```

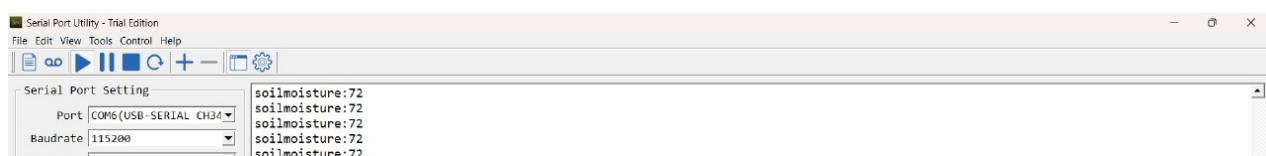
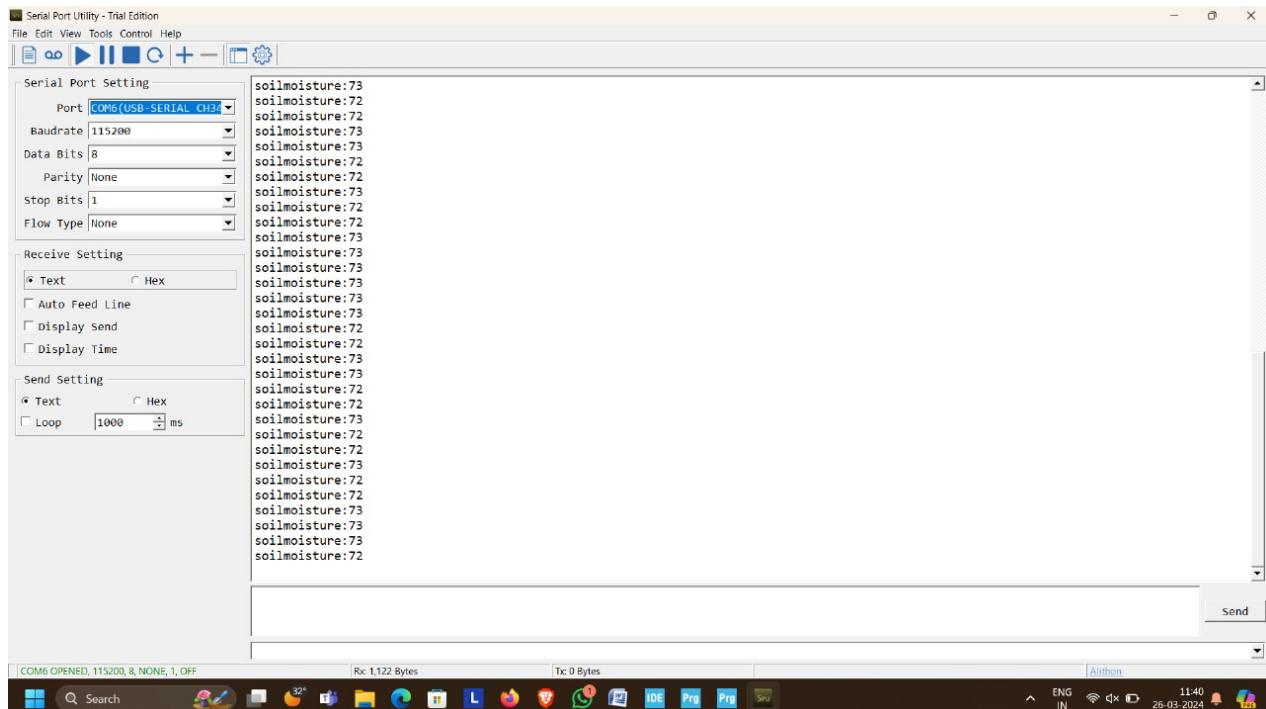
```

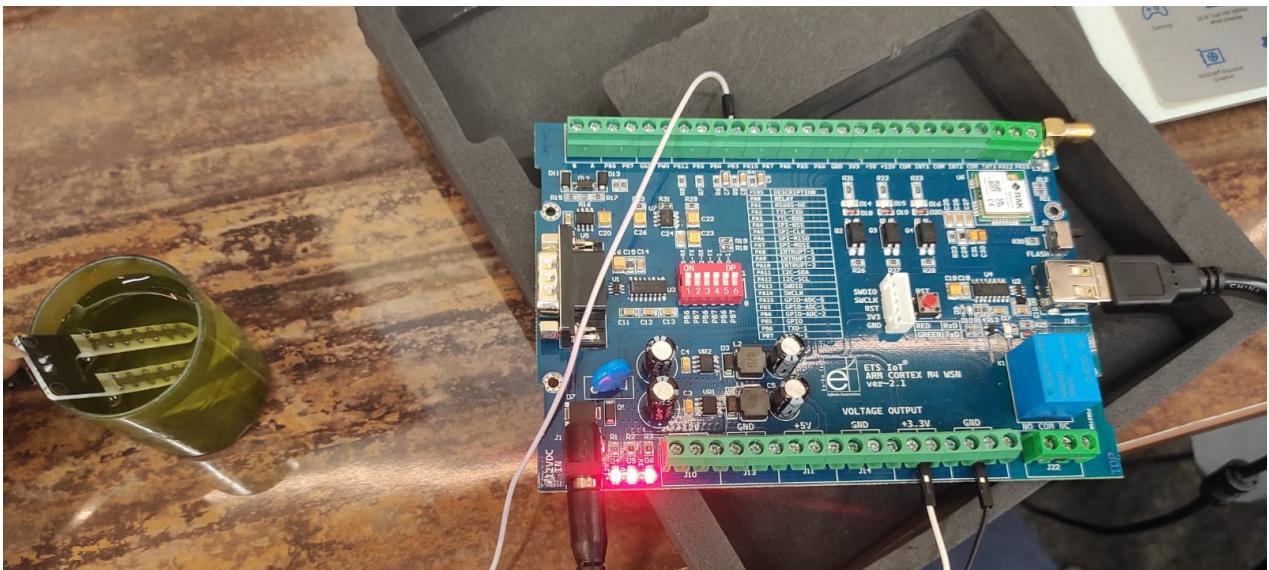
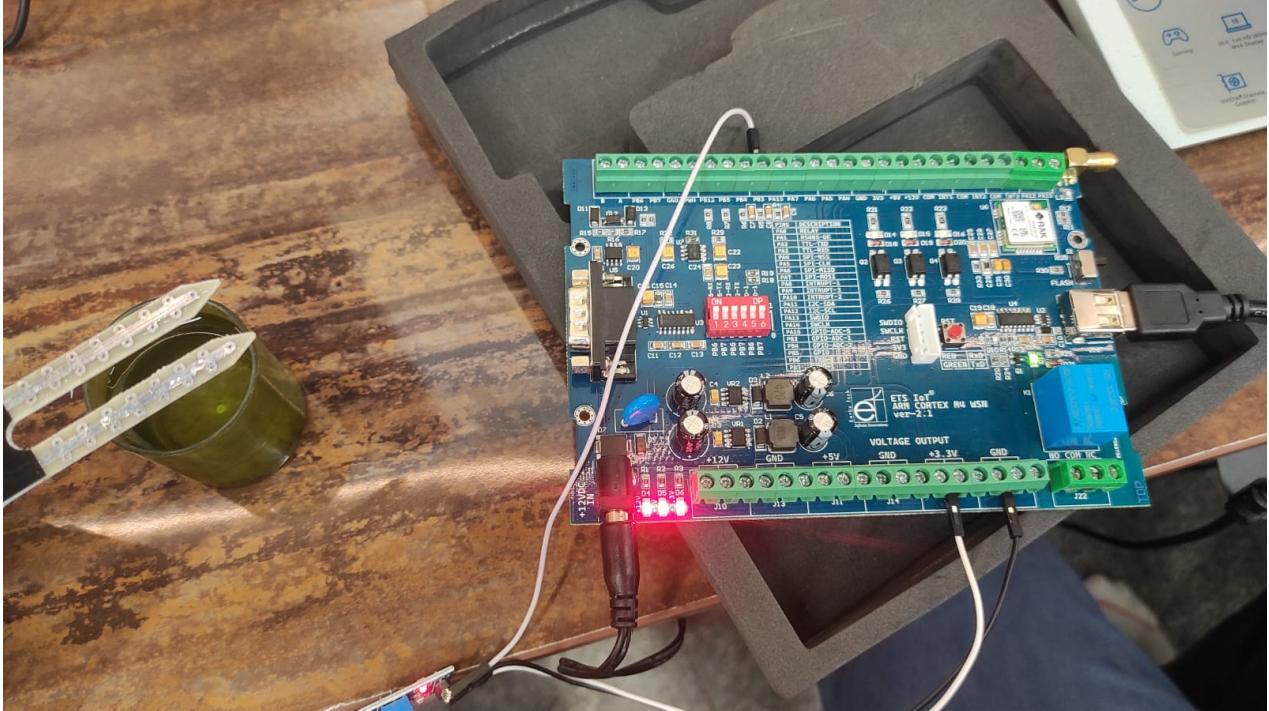
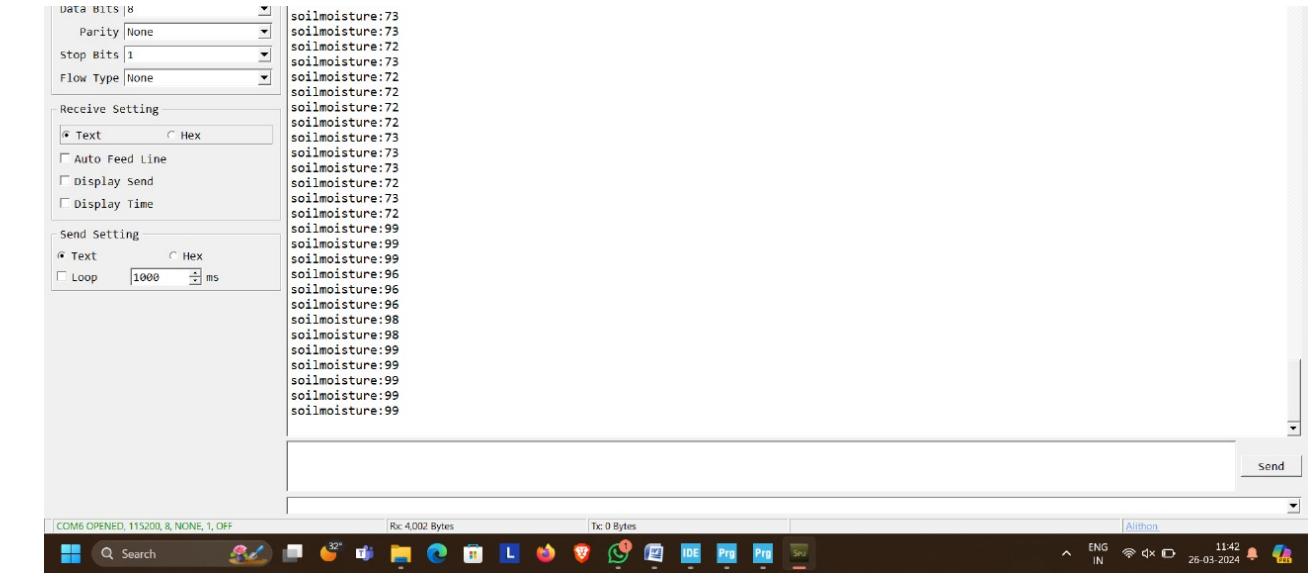
        return ch;
    }

while (1)
{
    HAL_ADC_Start(&hadc);
    HAL_ADC_PollForConversion(&hadc,100);
    adc_val = HAL_ADC_GetValue(&hadc);
    uint32_t soilmoisture;
    soilmoisture=adc_val/10.24;
    HAL_ADC_Stop(&hadc);
    HAL_Delay(500);
    printf("soilmoisture=%ld\n",soilmoisture);
    if(adc_val<500)
    {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0,
GPIO_PIN_RESET);
    }
    if(adc_val>500)
    {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0,
GPIO_PIN_SET);
    }
}

```

Output screen shots on serial monitor :







Result :

Interfacing a Analog Input (soil moisture sensor) with ARM microcontroller based IOT development is executed and the results visualized on serial monitor

Releases

No releases published

Packages

No packages published