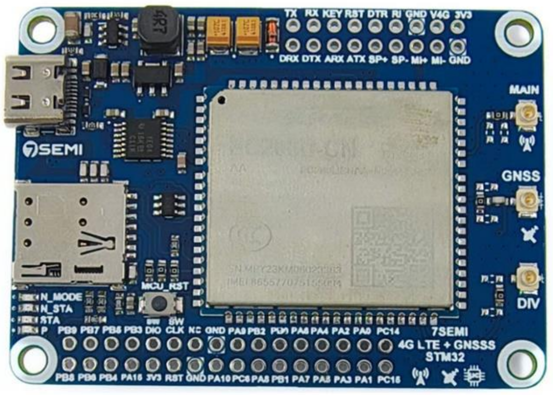
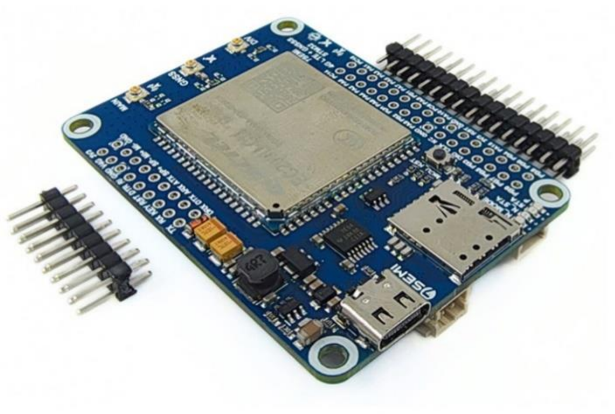
7SEMI

EC200U LTE 4G STM32

(Smart Modem Onboard STM32 MCU)

Version 1.0


TX RX KEY RST DTR RI GND V4G 3V3
DRX DTX ARX ATX SP+ SP- MI+ MI- GND
MAIN
GNSS
DIV
7SEMI
MCU_RST
N_MODE
N_STA
STA
P
7SEMI
4G LTE + GNSSS
STM32



7SEMI • +91 8655821342 • info@7semi.com • www.7semi.com

Table of Contents

1.Features

1. EC200U Series Module

Power Supply:

- Supply Voltage: 3.3–4.3 V.
- Typical Supply Voltage: 3.8 V.

Transmitting Power:

- Class 4 for EGSM850 and EGSM900.
- Class 1 for DCS1800 and PCS1900.
- Class 3 for LTE-FDD and LTE-TDD bands.

LTE Features:

- Supports Cat 1 FDD and TDD.
- RF Bandwidth: 1.4/3/5/10/15/20 MHz.
- FDD Data Rates: Max. 10 Mbps (DL) and Max. 5 Mbps (UL).
- TDD Data Rates: Max. 8.96 Mbps (DL) and Max. 3.1 Mbps (UL).

GSM Features:

- GPRS multi-slot class 12.
- Coding Schemes: CS-1 to CS-4.
- Max Data Rates: 85.6 kbps (DL and UL).

Internet Protocol Features:

- Supported Protocols: TCP, UDP, PPP, NTP, NITZ, FTP, HTTP, PING, CMUX, HTTPS, FTPS, SSL, FILE, MQTT, MMS.
- Authentication: PAP and CHAP for PPP connections.

SMS Features:

- Text and PDU modes.
- Point-to-Point MO and MT.
- Cell Broadcast.
- Storage: Stored in (U)SIM card and ME (default in ME).

SIM Interface:

- Supports 1.8/3.0 V SIM.
- Supports DSDS (Dual SIM Dual Standby).

Audio Features:

- Supports one analog audio input and one analog audio output.
- Supports HR/FR/EFR/AMR/AMR-WB.
- Features echo cancellation and noise suppression.

USB Interface:

- Compliant with USB 2.0 (slave mode only); data transfer rate up to 480 Mbps.
- Used for AT command communication, data transmission, software debugging, firmware upgrade.
- Supports USB serial drivers for Windows 7/8/8.1/10, Linux 2.6–5.12, and Android 4.x–11.x.

UART Interfaces:

- Main UART: Used for AT command communication, baud rate up to 921600 bps, supports hardware flow control.
- Debug UART: For Linux console and log output, baud rate 921600 bps.
- Auxiliary UART.

I2C Interfaces:

- Two I2C interfaces.

SPI Interface:

- Supports master mode only.

SD Card Interface:

- SD 2.0 protocol compliant.

WLAN Application Interface:

- Supports SDIO 1.1 interface for WLAN functionality.

USB_BOOT Interface:

- Forced download interface.

AT Commands:

- Compliant with 3GPP TS 27.007, 3GPP TS 27.005, and Quectel enhanced AT commands

Network Indication:

- NET_MODE and NET_STATUS for network connectivity indication.

Antenna Interfaces:

- Main antenna (ANT_MAIN).
- Wi-Fi/Bluetooth antenna (ANT_BT/WIFI_SCAN).
- GNSS antenna (ANT_GNSS).

Location Support:

- Supports Wi-Fi Scan and GNSS.

Physical Characteristics:

- Size: 28.0 mm × 31.0 mm × 2.4 mm.
- Weight: Approx. 4.1 g.

Temperature Ranges:

- Operating Temperature: -35 °C to +75 °C.
- Extended Temperature: -40 °C to +85 °C.
- Storage Temperature: -40 °C to +90 °C.

Firmware Upgrade:

- Via USB interface or FOTA (Firmware Over-The-Air).

RoHS Compliance:

- Fully compliant with EU RoHS directive.

2. STM32

CPU & Memory:

- Core: Arm 32 bit Cortex-M0+ CPU running at up to 64 MHz.
- Flash Memory: 128 KB.
- SRAM: 32 KB.
- Debug Interface: SWD (Serial Wire Debug).

Power Management:

- Operating Voltage: 2.0V to 3.6V.
- Low-power modes: Sleep, Stop, Standby

Peripherals:

- **GPIOs:** 29 general-purpose I/O pins.
- **Timers:**
    - 11 timers: 16-bit for advanced motor control, five 16-bit general-purpose, two basic 16-bit, two watchdogs, SysTick timer
- **Communication Interfaces:**
    - 4x USART, 2x I2C, 2x SPI.
- **Analog:**
    - 12-bit ADC with up to 16 external channels.
- **Clock management**
    - 4 to 48 MHz crystal oscillator
    - 32 kHz crystal oscillator with calibration
    - Internal 16 MHz RC with PLL option
    - Internal 32 kHz RC oscillator (±5 %)

1.1Description

This **7semi EC200U** and **STM32 board** integrates both the EC200U LTE module and the STM32 microcontroller on a single platform, offering a powerful solution for IoT and communication-based applications.

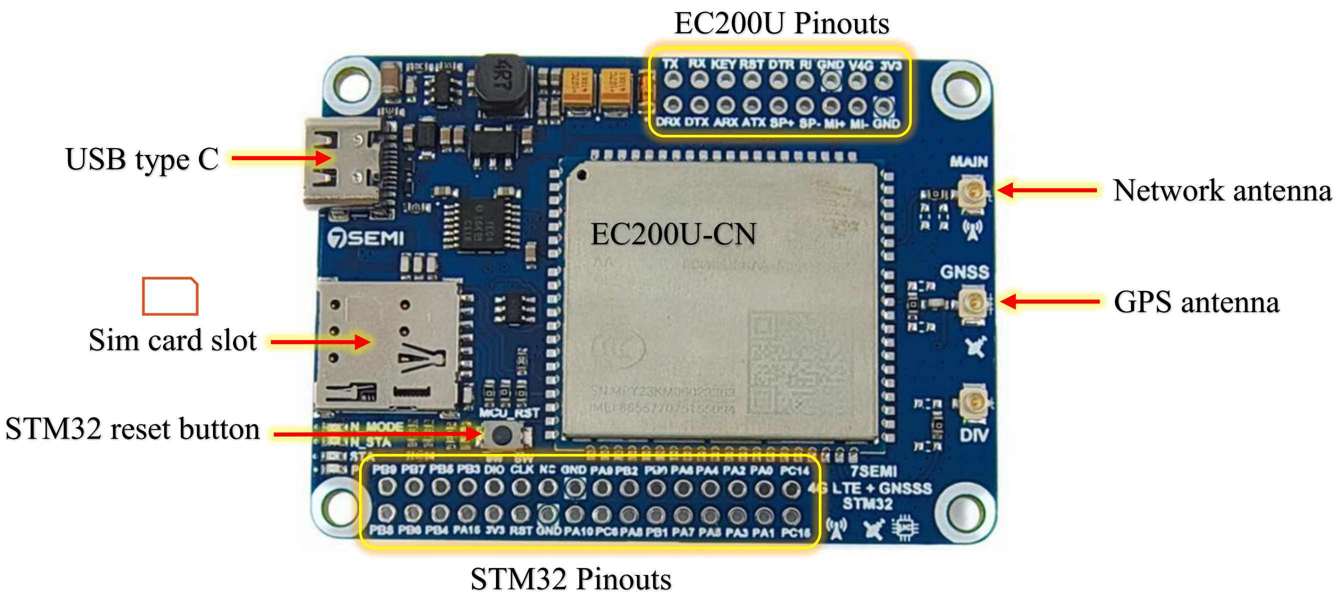
- **EC200U Module**: A 4G LTE Cat 1 modem, providing cellular communication with support for GPS/GNSS, enabling long-distance data transmission and location tracking. It offers high-speed internet over LTE networks and supports fallback to GSM, making it versatile for different regions and network conditions. The EC200U module also supports multiple communication protocols like TCP/IP, FTP, and MQTT, suitable for IoT deployments.
- **STM32 Module**: The STM32G070KBT6 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.

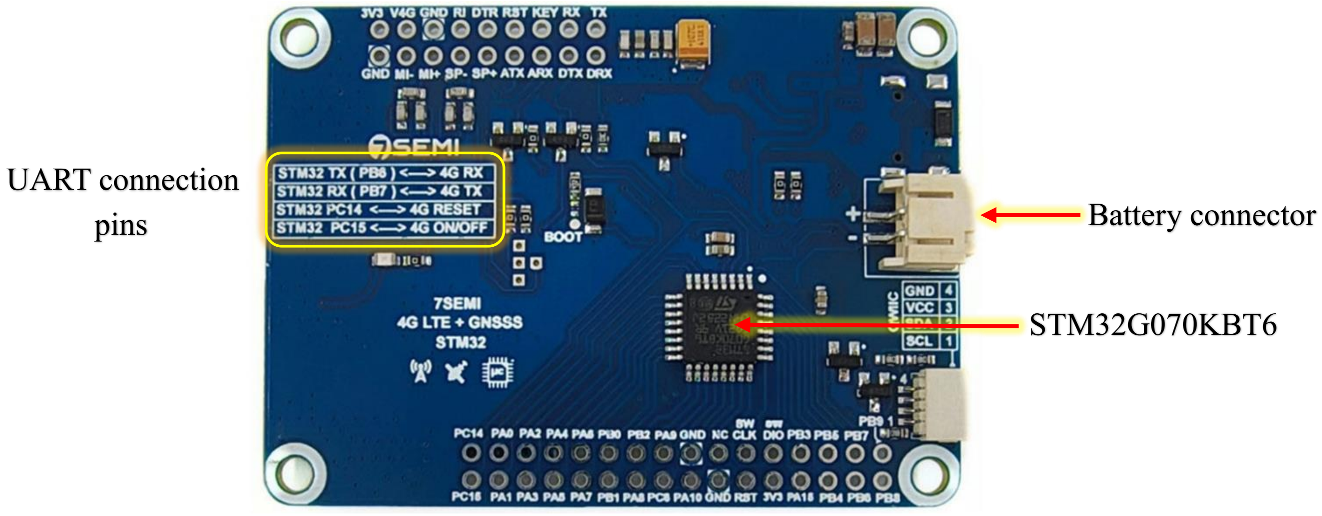
Combined Features:

- **Seamless Communication**: The EC200U and STM32 communicate effectively through UART, allowing control of cellular functions (like SMS, internet access) from the STM32 without external wiring.
- **Power Management**: Both modules are optimized for low power consumption, making the board ideal for battery-powered IoT applications.
- **Peripheral Sharing**: The board allows shared use of peripherals like antennas and communication buses, optimizing design space and performance.

This integrated board is ideal for IoT, industrial automation, GPS tracking, and remote data monitoring applications.

1.2Pinouts

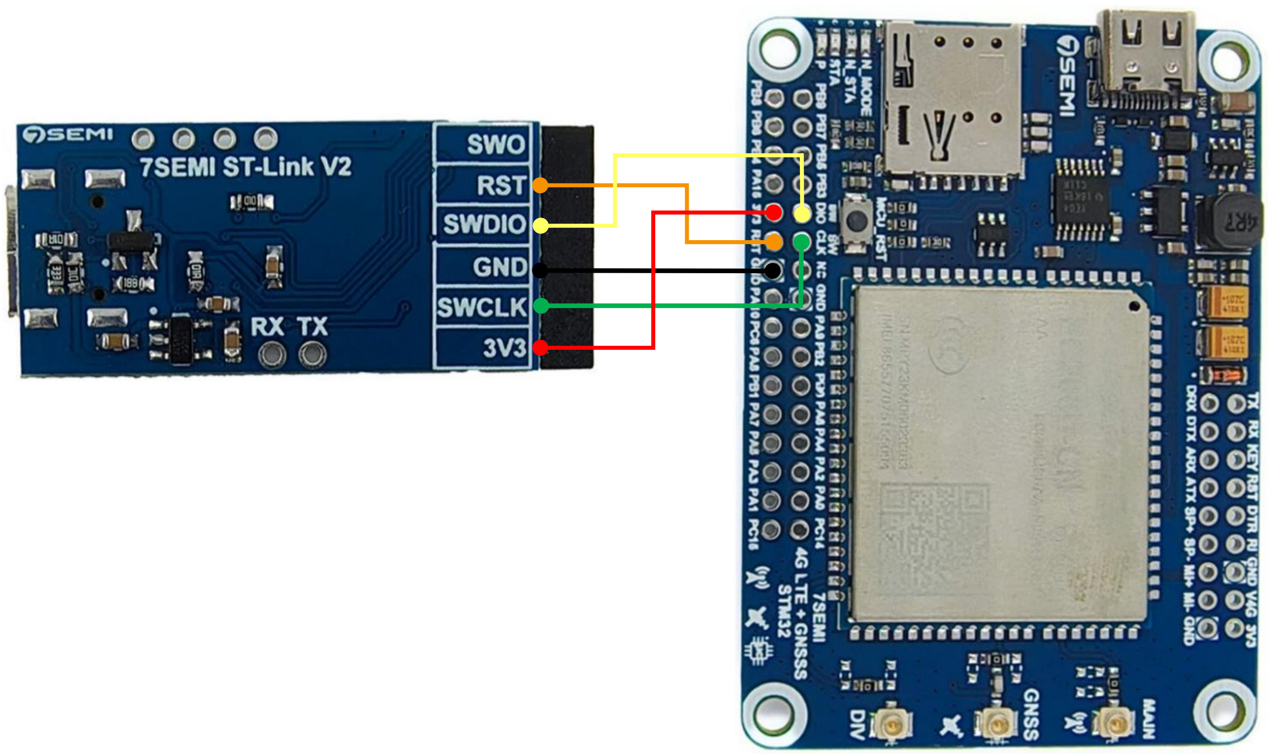

EC200U Pinouts
TX RX KEY RST DTR RI GND V4G 3V3
DRX DTX ARX ATX SP+ SP- MI+ MI- GND
USB type C
Network antenna
EC200U-CN
GPS antenna
Sim card slot
STM32 reset button
PB9 PB7 PB5 PB3 DIO CLK NC GND PA9 PB2 PB0 PA6 PA4 PA2 PA0 PC14
PB8 PB6 PB4 PA15 3V3 RST GND PA10 PC6 PA5 PB1 PA7 PA5 PA3 PA1 PC15
STM32 Pinouts


UART connection pins
STM32 TX (PB6) <—> 4G RX
STM32 RX (PB7) <—> 4G TX
STM32 PC14 <—> 4G RESET
STM32 PC15 <—> 4G ON/OFF
Battery connector
STM32G070KBT6

2.Connection of 7Semi EC200U STM32 Board with ST-link

To program this board with **STM32cubeIDE** software we need **7Semi ST-LINK V2/V2.1 USB-C Debugger Programmer**.

Programmer link:- https://shorturl.at/qY4c4



Note:- If you want to first test some basic AT commands with **only EC200U-CN** module then you can connect Type C cable directly to the PC and start sending AT commands using Serial terminal such as ***Coolterm***, ***Putty***, ***Docklight*** etc.

2.1 Connection Explain of board to Audio Amplifier and Speaker for Calling

In this setup, the 7Semi EC200U and STM32 modules are integrated on the same board, with the EC200U module responsible for managing cellular communication, while the STM32 manages processing tasks and peripheral control. Here's how you can connect the board to an audio amplifier and speaker for voice calling functionality using AT commands:

Components Involved:

1. **7Semi EC200U Module**: Responsible for LTE communication and voice calling.
2. **STM32 Module**: Acts as the controller to issue AT commands to the EC200U.
3. **Audio Amplifier**: Amplifies the audio signal from the EC200U for the speaker.
4. **Speaker**: Outputs the audio during a call.
5. **Microphone (optional)**: For voice input during calls.
6. **SIM Card**: Inserted in the EC200U's SIM slot for cellular network access.

Connection Overview:

1. **STM32 to EC200U**:
   - **UART Interface**: The STM32 communicates with the EC200U using UART (RX/TX). This allows the ESP32 to send AT commands for making calls, sending SMS, and managing data over LTE.
   - **Power Supply**: Ensure both modules share the correct voltage supply (3.3V or 3.8V for the EC200U as specified).
   -
2. **EC200U to Audio Amplifier**:
   - The EC200U has **audio input/output (analog)** pins that can be connected to the amplifier. These are used to transmit the voice signals during a call.
   - **Amplifier**: Connect the audio output from the EC200U to the amplifier's input.
   - **Speaker**: The amplifier's output is then connected to the speaker for audio playback.
3. **STM32 Control**:
   - The STM32 sends **AT commands** to the EC200U.

Example AT Commands for Calling:

- **Initiating a Call**:

```
ATD+1234567890;  // Dials the number 1234567890
```

- **Answering a Call**:

```
ATA  // Answers an incoming call
```

- **Hanging Up a Call**:

```
ATH  // Ends the current call
```

- **Controlling Audio Output**: Use specific AT commands to route audio to the correct output (e.g., speaker), control volume, and manage audio settings during a call.

Additional Features:

- **Speaker Control**: The ESP32 can adjust the audio settings (volume, mute, etc.) by sending AT commands through the UART interface.
- **Microphone Integration**: If a microphone is integrated, the EC200U can handle both incoming and outgoing audio streams, enabling two-way voice communication.

This configuration provides a fully functional calling solution with audio playback through a speaker, controlled by the ESP32 module, which sends AT commands to the EC200U to manage the call.

Pin Connection Explanation for Amplifier and Speaker Setup:

1. SP+ and SP- (EC200U Audio Output):

- The **SP+** and **SP-** pins on the **EC200U module** provide the **analog audio output**. These pins are designed to carry the voice signal during a call.
- **SP+**: Carries the positive audio signal.
- **SP-**: Carries the negative (grounded) audio signal.

These audio signals need to be amplified to drive the speaker.

2. Connecting SP+ and SP- to the Amplifier Input:

- The **SP+** pin is connected to the **Input+** pin of the amplifier.
- The **SP-** pin is connected to the **Input-** pin of the amplifier.

This configuration allows the amplifier to receive the differential audio signal from the EC200U module.

3. Speaker Connection to Amplifier Output:

- The speaker is connected to the **Output+** and **Output-** pins of the amplifier board:
- **Output+** from the amplifier goes to the **positive terminal** of the speaker.
- **Output-** from the amplifier goes to the **negative terminal** of the speaker.

The amplifier boosts the audio signal coming from the EC200U and drives the speaker for clear voice output during a call.

4. Powering the Amplifier:

- The amplifier requires a power supply, which should be connected to the **power input terminals** of the amplifier board.
- The required input voltage is typically specified by the amplifier board (e.g., 5V, 12V, depending on the amplifier's design).
- Make sure the input voltage is stable and within the range recommended by the amplifier manufacturer to avoid damage or poor performance.

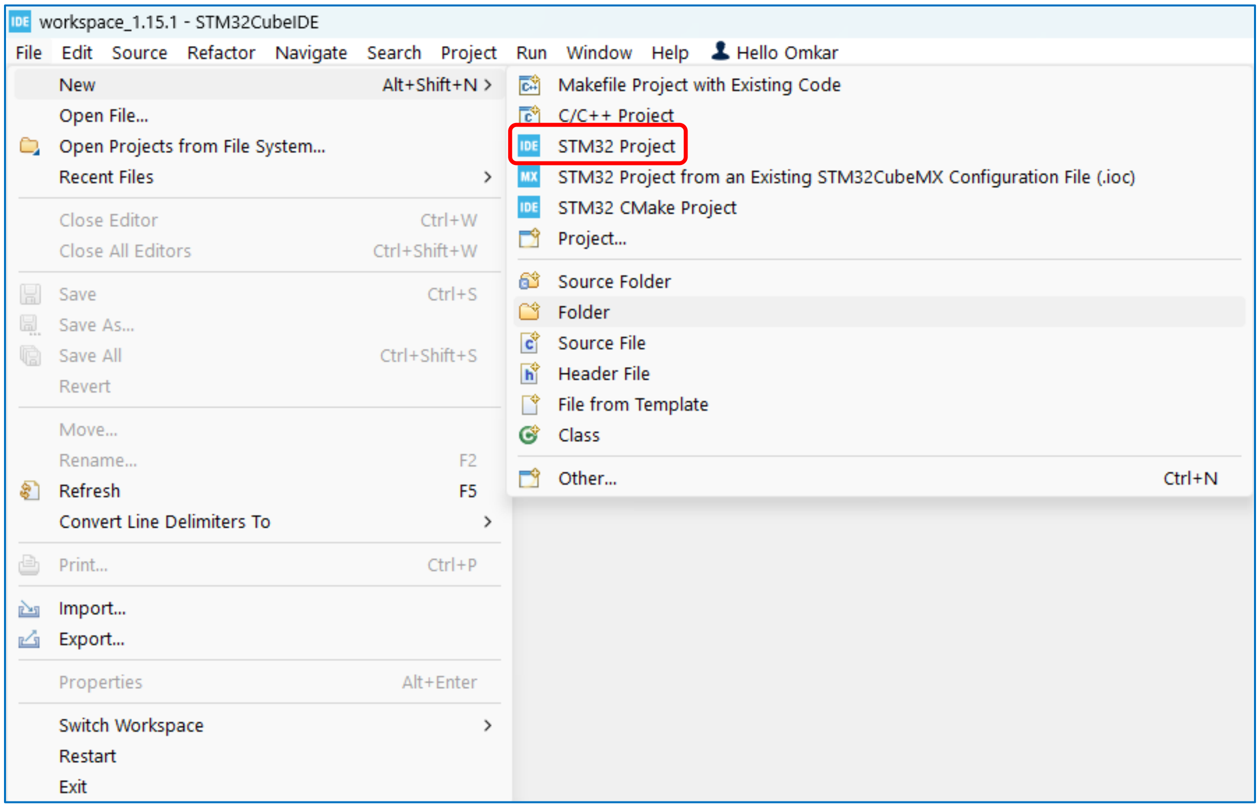
Summary of Connections:

- **SP+** (EC200U) → **Input+** (Amplifier).
- **SP-** (EC200U) → **Input-** (Amplifier).
- **Output+** (Amplifier) → **Speaker +**.
- **Output-** (Amplifier) → **Speaker -**.
- **Power Supply**: Connect to the amplifier’s power input for operation.

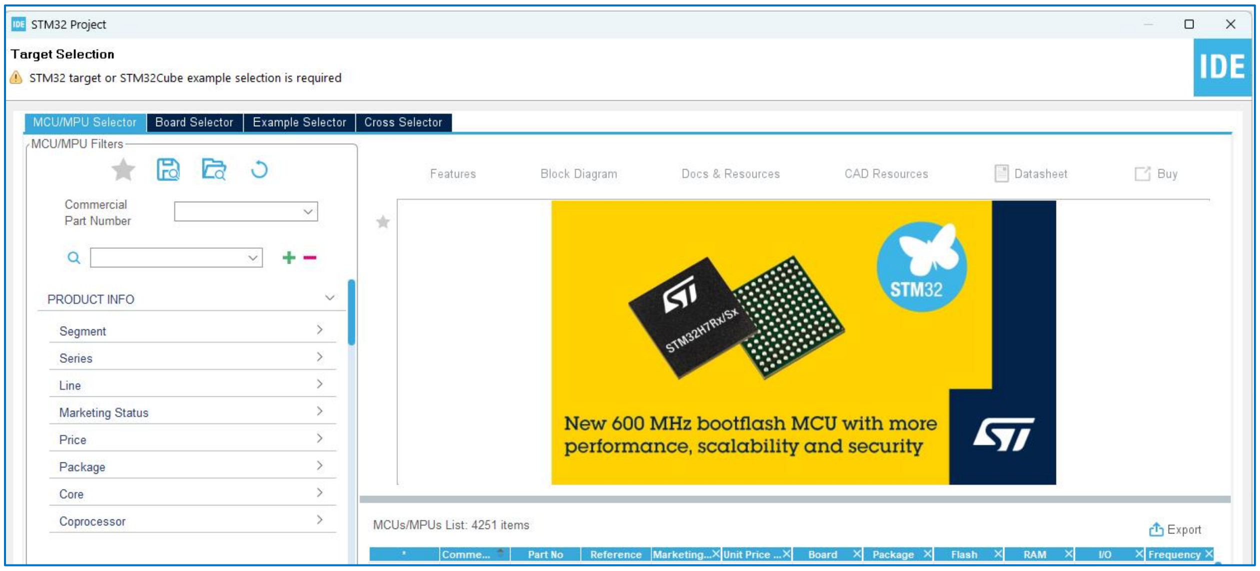
This setup ensures that the audio signal from the EC200U is amplified and output through the speaker for clear sound during a call.

2.2 Getting started with STM32 project to send SMS

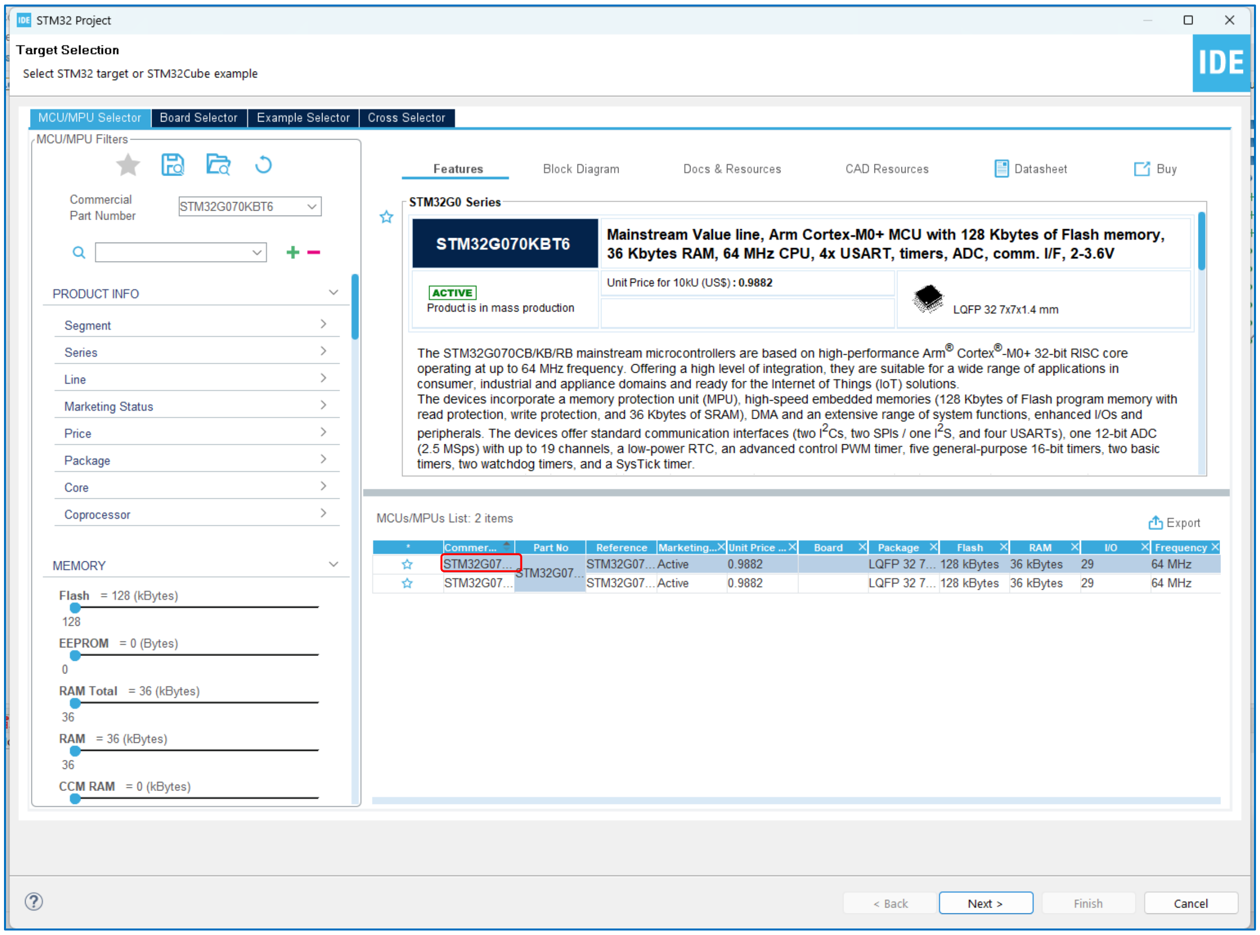
Step 1:- Open STM32 cube IDE, Go to ***File*** > ***New*** > ***STM32 project***



Step 2:- Enter the part number of the STM32 which we are using, ***MCU/MPU selector*** > ***Commercial part number*** (STM32G070KBT6)



Select on the first option and click ***Next*** at the bottom.



Target Selection

Select STM32 target or STM32Cube example

Commercial Part Number: STM32G070KBT6

Flash = 128 (kBytes)
128

EEPROM = 0 (Bytes)
0

RAM Total = 36 (kBytes)
36

RAM = 36 (kBytes)
36

CCM RAM = 0 (kBytes)

STM32G0 Series

STM32G070KBT6

Mainstream Value line, Arm Cortex-M0+ MCU with 128 Kbytes of Flash memory, 36 Kbytes RAM, 64 MHz CPU, 4x USART, timers, ADC, comm. I/F, 2-3.6V

ACTIVE
Product is in mass production

Unit Price for 10kU (US$) : 0.9882

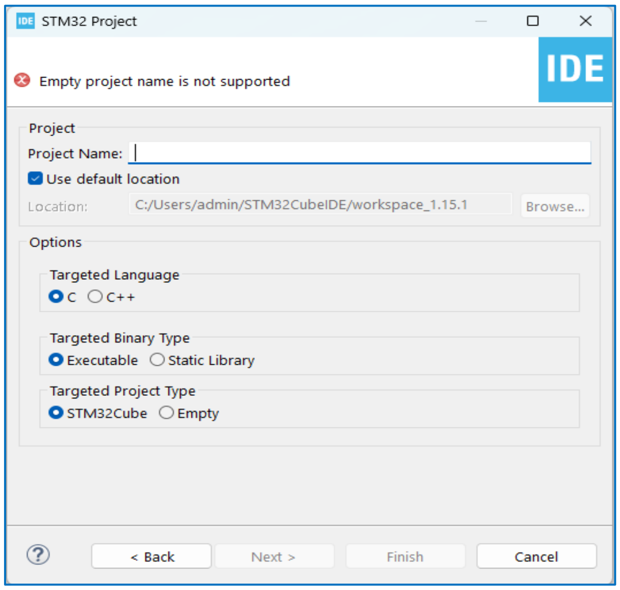
LQFP 32 7x7x1.4 mm

The STM32G070CB/KB/RB 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 (128 Kbytes of Flash program memory with read protection, write protection, 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 I^{2}Cs, two SPIs / one I^{2}S, and four USARTs), one 12-bit ADC (2.5 MSps) with up to 19 channels, a low-power RTC, an advanced control PWM timer, five general-purpose 16-bit timers, two basic timers, two watchdog timers, and a SysTick timer.

MCUs/MPUs List: 2 items

*	Commer...	Part No	Reference	Marketing...	Unit Price ...	Board	Package	Flash	RAM	I/O	Frequency
☆	STM32G07...	STM32G07...	STM32G07...	Active	0.9882		LQFP 32 7...	128 kBytes	36 kBytes	29	64 MHz
☆	STM32G07...		STM32G07...	Active	0.9882		LQFP 32 7...	128 kBytes	36 kBytes	29	64 MHz

Step 3:- Give your project a name and click.



Empty project name is not supported

Project

Project Name:

[✓] Use default location

Location: C:/Users/admin/STM32CubeIDE/workspace_1.15.1

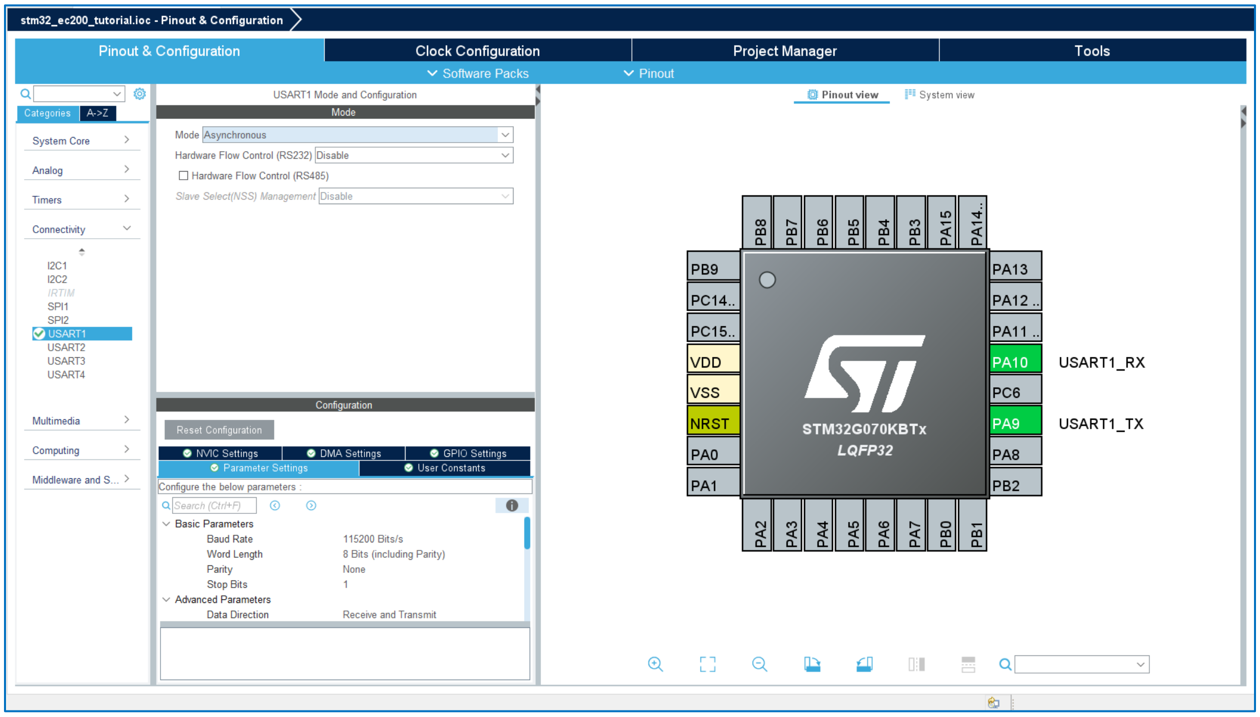
Options

Targeted Language
[✓ C] C++

Targeted Binary Type
[✓ Executable] Static Library

Targeted Project Type
[✓ STM32Cube] Empty

Step 4:- Open ***.ioc*** file from project explorer from left side corner, Under ***Pinout & Configuration > Connectivity > USART1 > Mode > Asynchronous***

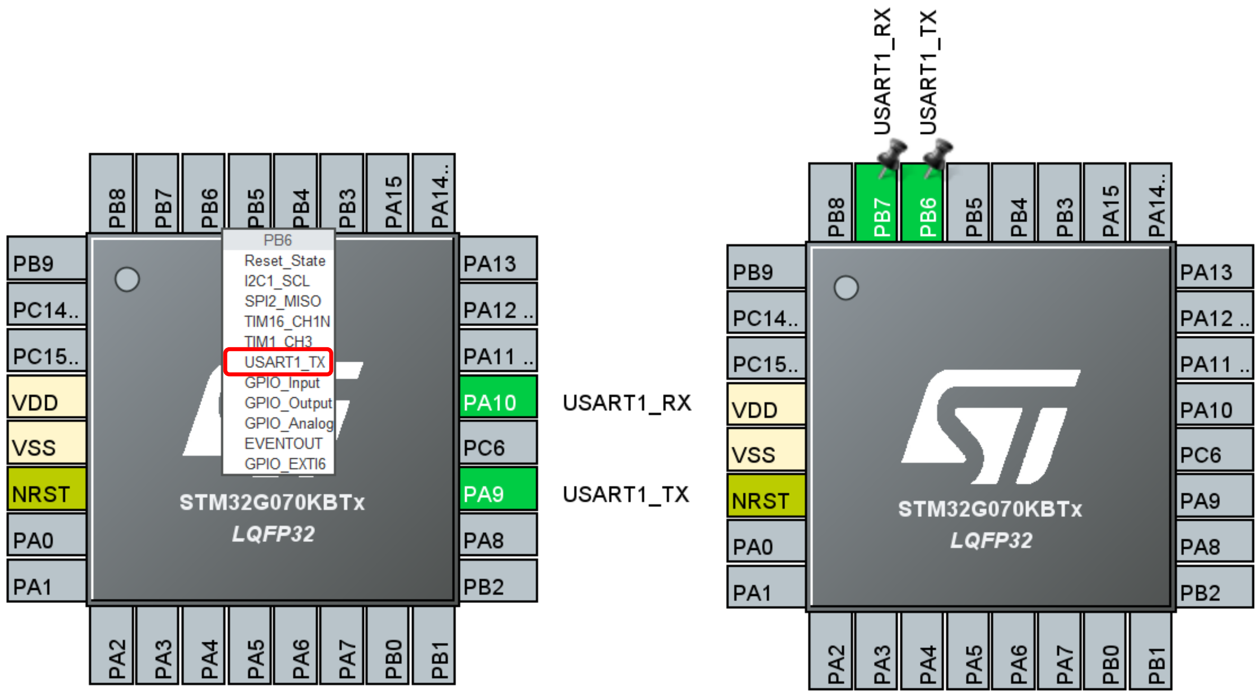


Mode: Asynchronous
Hardware Flow Control (RS232): Disable
☐ Hardware Flow Control (RS485)
Slave Select(NSS) Management: Disable

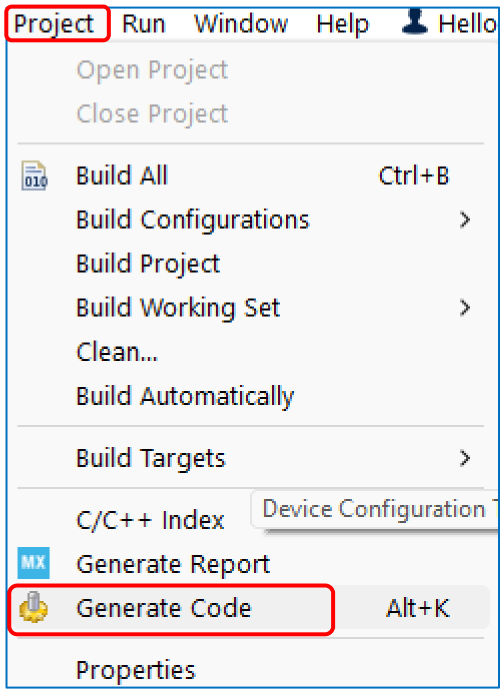
Configure the below parameters :

Basic Parameters	
Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit

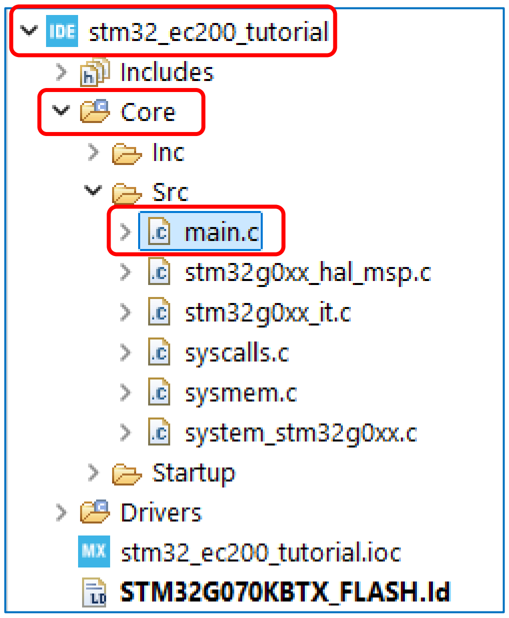
We need to set ***PB6*** as ***USART1_TX*** and ***PB7*** as ***USART1_RX***, click on ***PB6*** and select ***USART1_TX*** and for ***PB7*** select ***USART1_RX***



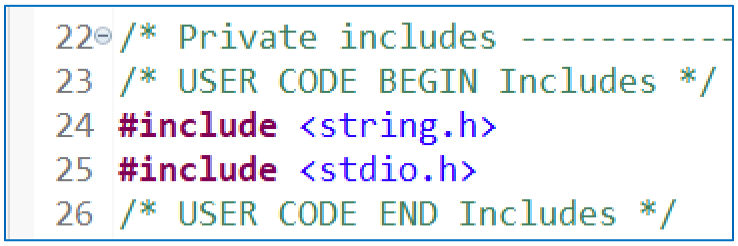
Step 5:- Go to ***Project*** > ***Generate code***



Step 6:- Go to your ***Project name*** > ***Core*** > ***Src*** > ***main.c***



Step 7:- Copy and paste these two lines as shown in the image.



```
/* Private includes -----------
/* USER CODE BEGIN Includes */
#include <string.h>
#include <stdio.h>
/* USER CODE END Includes */
```

```
#include <string.h>
#include <stdio.h>
```

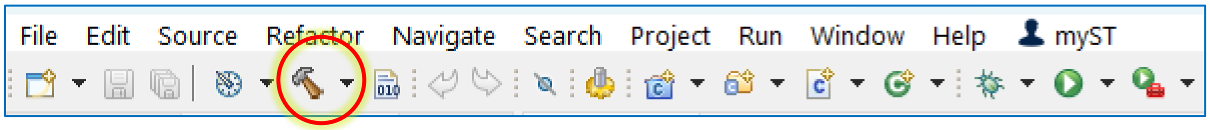
Copy from here

After that paste the code given below inside ***/* USER CODE BEGIN 2 */***

```
char mobileNumber[] = "+91xxxxxxxxxx";  // Enter the Mobile Number you want to send to
   char ATcommand[80];
   uint8_t buffer[30] = {0};
   uint8_t ATisOK = 0;
   while(!ATisOK){
               sprintf(ATcommand,"AT\r\n");
               HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
               HAL_UART_Receive (&huart1, buffer, 30, 100);
               HAL_Delay(1000);
               if(strstr((char *)buffer,"OK")){
                       ATisOK = 1;
               }
               HAL_Delay(1000);
               memset(buffer,0,sizeof(buffer));
   }
```

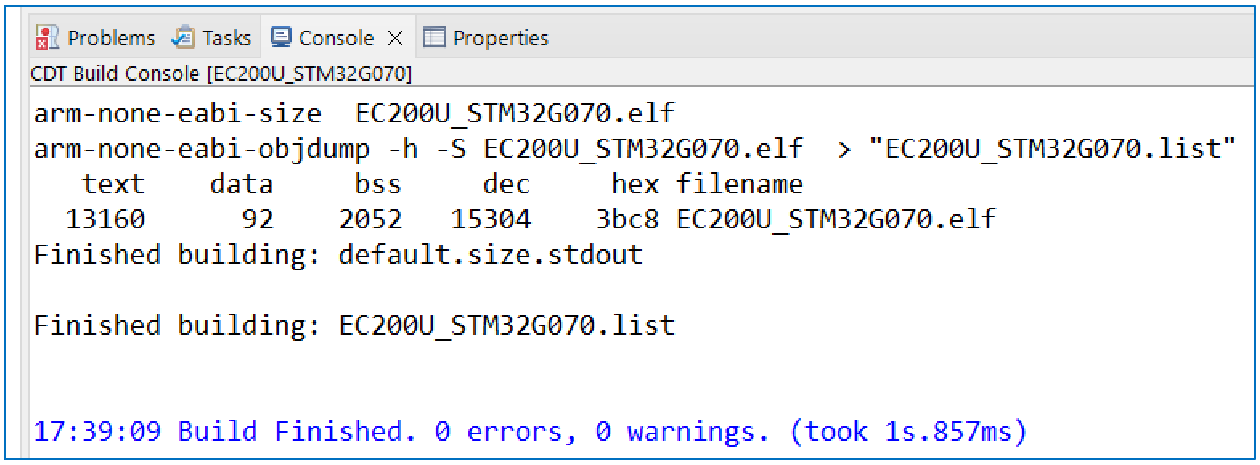
```
sprintf(ATcommand,"AT+CMGF=1\r\n");
    HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
    HAL_UART_Receive (&huart1, buffer, 30, 1000);
    //The data received from EC200U is stored in "buffer", then print on serial what you
received
    //HAL_UART_Transmit(&huart1,(uint8_t *)buffer,strlen((char*)buffer),1000);
    HAL_Delay(2000);


    memset(buffer,0,sizeof(buffer)); //Clear the buffer to store new data


    sprintf(ATcommand,"AT+CMGS=\"%s\"\r\n",mobileNumber);
    HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
    HAL_UART_Receive (&huart1, buffer, 30, 1000);
    //The data received from EC200U is stored in "buffer", then print on serial what you
received
    //HAL_UART_Transmit(&huart1,(uint8_t *)buffer,strlen((char*)buffer),1000);
    HAL_Delay(2000);
    memset(buffer,0,sizeof(buffer)); //Clear the buffer to store new data


    sprintf(ATcommand,"Hello World, STM32 started%c",0x1a); //0x1a is Ctrl+Z
    HAL_UART_Transmit(&huart1,(uint8_t *)ATcommand,strlen(ATcommand),1000);
    HAL_UART_Receive (&huart1, buffer, 30, 100);
    //The data received from EC200U is stored in "buffer", then print on serial what you
received
    //HAL_UART_Transmit(&huart1,(uint8_t *)buffer,strlen((char*)buffer),1000);
    memset(buffer,0,sizeof(buffer));
    HAL_Delay(4000);
```

Step 8:- Now let's ***build*** this project and check for any errors.



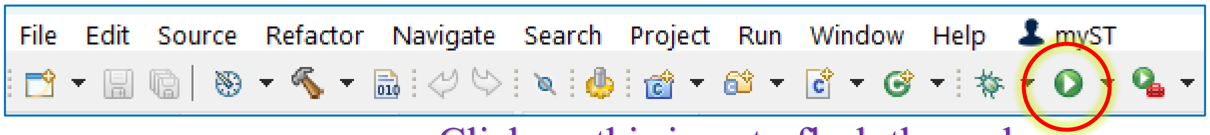
Click on this icon to build the project

You should get message similar to which is shown below



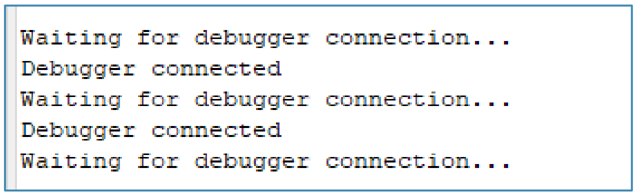
```
CDT Build Console [EC200U_STM32G070]
arm-none-eabi-size  EC200U_STM32G070.elf
arm-none-eabi-objdump -h -S EC200U_STM32G070.elf  > "EC200U_STM32G070.list"
   text    data     bss     dec     hex filename
  13160      92    2052   15304    3bc8 EC200U_STM32G070.elf
Finished building: default.size.stdout

Finished building: EC200U_STM32G070.list


17:39:09 Build Finished. 0 errors, 0 warnings. (took 1s.857ms)
```

Step 9:- Now it's time to upload the code into the STM32 microcontroller.

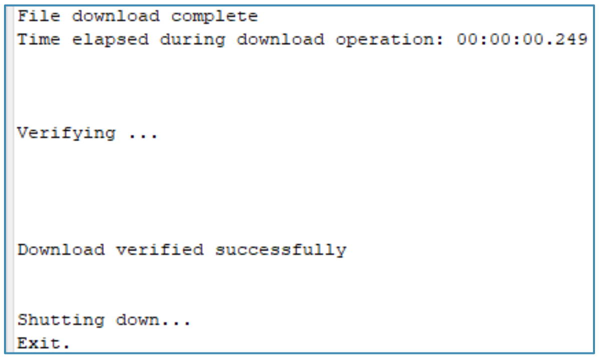


Click on this icon to flash the code

After successful upload of the code you will get to see the message as shown below in the console



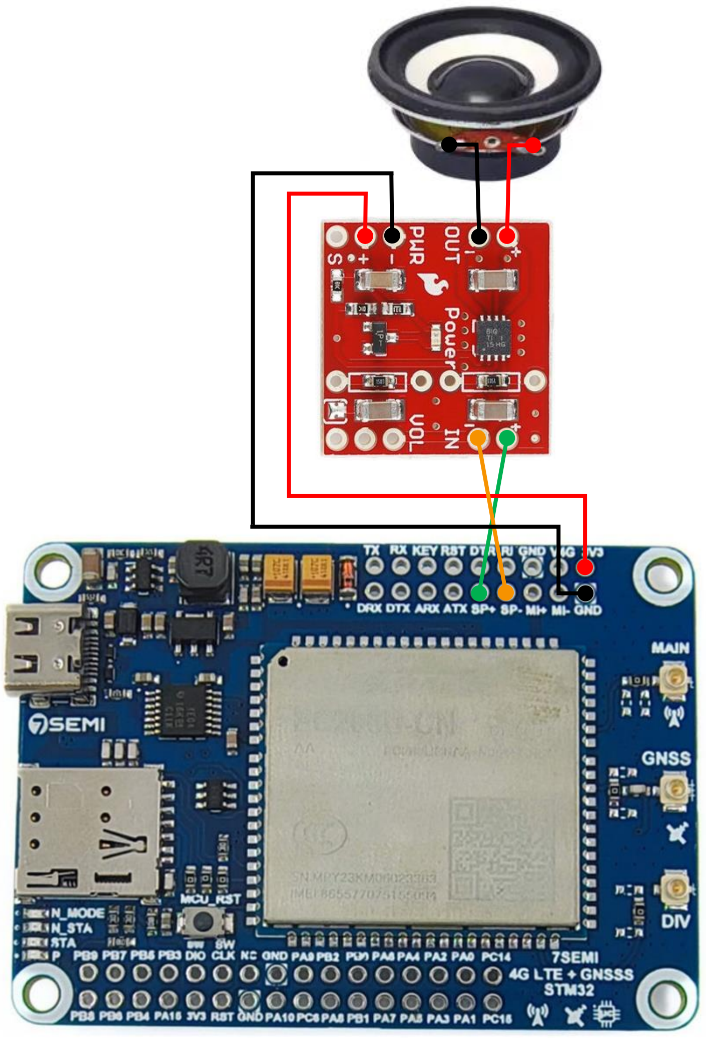
```
Waiting for debugger connection...
Debugger connected
Waiting for debugger connection...
Debugger connected
Waiting for debugger connection...
```



```
File download complete
Time elapsed during download operation: 00:00:00.249



Verifying ...



Download verified successfully


Shutting down...
Exit.
```

After successfully uploading the code into STM32 board, press the STM32 restart button. You should receive the text message to the number which you have mentioned.

2.3 Connection of the board with speaker



3.Contact Information

7Semi is a leading provider of wide range of efficient and accessible hardware products and related technical solutions to an extensive range of industries like IoT, Automation, Education and Learning, Robotics and more

Call us at	: +91 8655821342
Email us at	: info@7semi.com

We're happy to answer questions.

If you need assistance – Click here to more