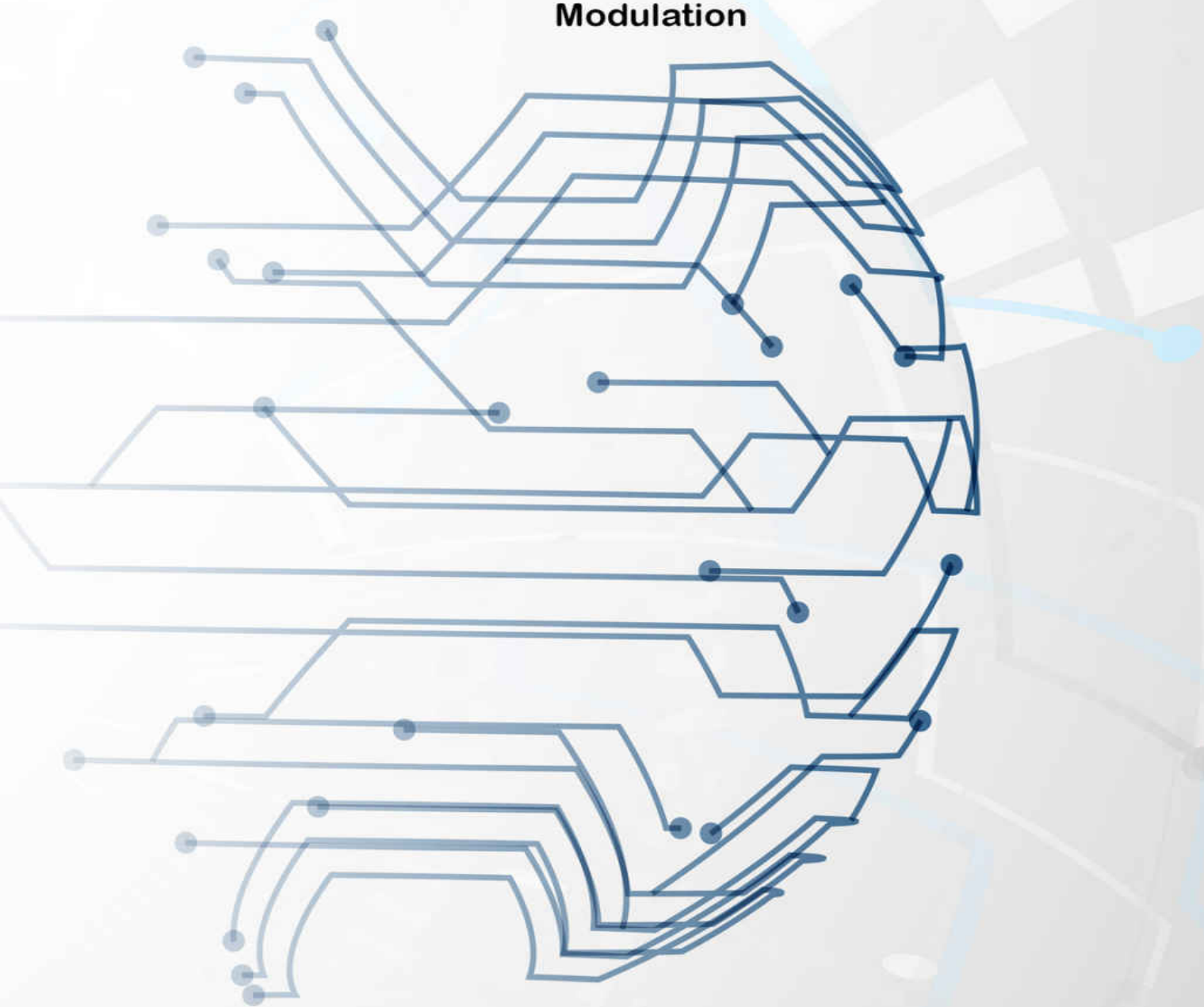
Programming STM32 Microcontroller circuit

STM32 Microcontroller, Keil uVision and STM32CubeMX, ESP8266 with STM32F103C8,Stepper and Servo Motor with STM32F103C8, Heartbeat width Modulation



Programming STM32 Microcontroller circuit projects hand on

STM32 Microcontroller, Keil uVision and STM32CubeMX, ESP8266 with STM32F103C8,Stepper and Servo Motor with STM32F103C8, Heartbeat width Modulation..etc..,

Anbazhagan K

Contents

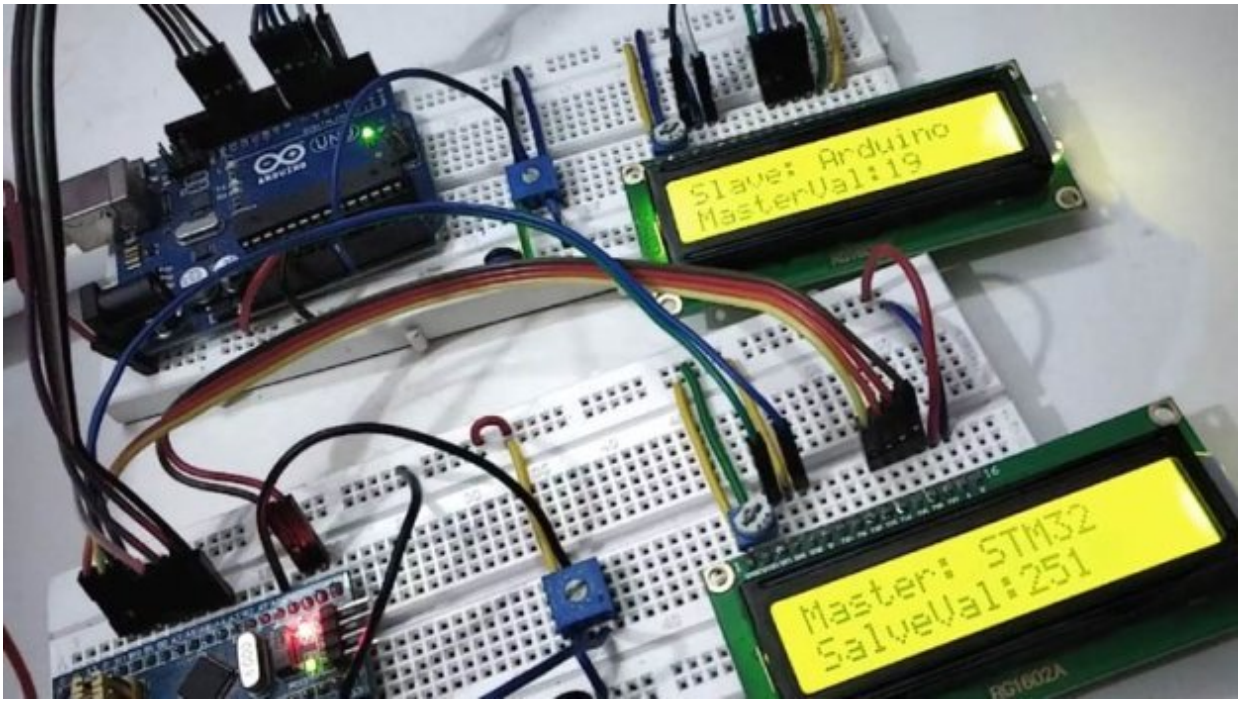
Acknowledgments

The writer might want to recognize the diligent work of the article group in assembling this book. He might likewise want to recognize the diligent work of the Raspberry Pi Foundation and the Arduino bunch for assembling items and networks that help to make the Internet of Things increasingly open to the overall population. Yahoo for the democratization of innovation!

Introduction

The Internet of Things (IOT) is a perplexing idea comprised of numerous PCs and numerous correspondence ways. Some IOT gadgets are associated with the Internet and some are most certainly not. Some IOT gadgets structure swarms that convey among themselves. Some are intended for a solitary reason, while some are increasingly universally useful PCs. This book is intended to demonstrate to you the IOT from the back to front. By structure IOT gadgets, the per user will comprehend the essential ideas and will almost certainly develop utilizing the rudiments to make his or her very own IOT applications. These included ventures will tell the per user the best way to assemble their very own IOT ventures and to develop the models appeared. The significance of Computer Security in IOT gadgets is additionally talked about and different systems for protecting the IOT from unapproved clients or programmers. The most significant takeaway from this book is in structure the tasks yourself.

1. Step by step instructions to Use SPI Communication in STM32 Microcontroller



In our past instructional exercises, we have found out about SPI and I2C correspondence between two Arduino sheets. In this instructional exercise we will supplant one Arduino board with the Blue Pill board that is STM32F103C8 and will speak with the Arduino board utilizing SPI transport.

In this STM32 SPI Example, we will utilize Arduino UNO as Slave and STM32F103C8 as Master with Two 16X2 LCD show joined to one another independently. Two Potentiometers are likewise associated with STM32 (PA0) and Arduino (A0) to decide the sending esteems (0 to 255) from ace to slave and slave to ace by fluctuating the potentiometer.

SPI in STM32F103C8

Looking at SPI transport in Arduino and STM32F103C8 Blue Pill board, STM32 has 2 SPI transport in it while Arduino Uno has one SPI transport. Arduino Uno has ATMEGA328 microcontroller in it, and STM32F103C8 has ARM Cortex-M3 which makes it quicker than Arudino Board.

To study SPI correspondence, allude our past articles

- Step by step instructions to utilize SPI in Arduino: Communication between two Arduino Boards

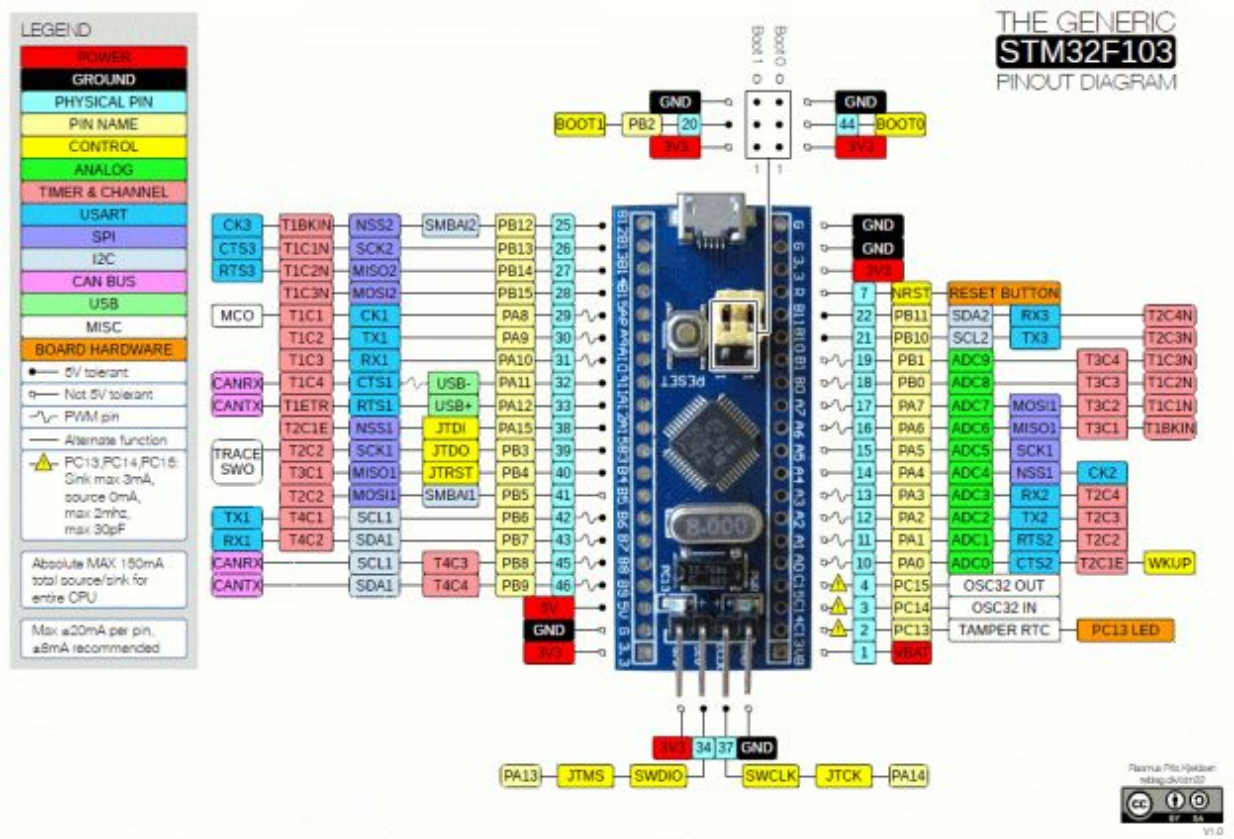
- SPI Communication with PIC Microcontroller PIC16F877A

- SPI correspondence by means of Bit Banging

- Raspberry Pi Hot Water Tank Leak Detector utilizing SPI Modules

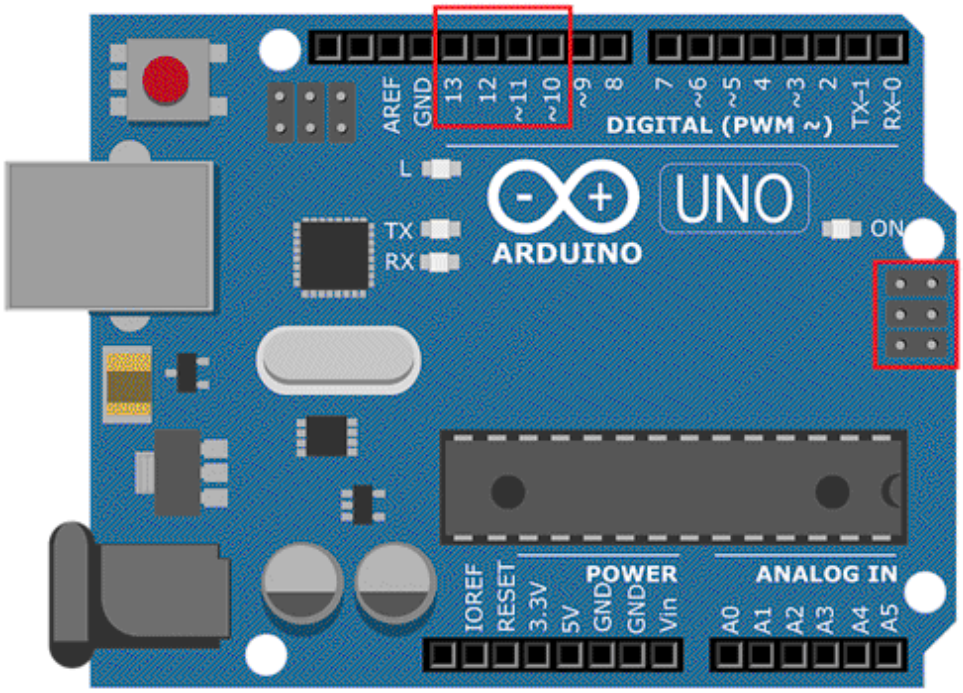
- ESP32 Real Time Clock utilizing DS3231 Module

SPI Pins STM32F103C8



SPI Line1	Pin in STM32F103C8
MOSI1	PA7 or PB5
MISO1	PA6 or PB4
SCK1	PA5 or PB3
SS1	PA4 or PA15
SPI Line2	
MOSI2	PB15
MISO2	PB14
SCK2	PB13
SS2	PB12

SPI Pins in Arduino



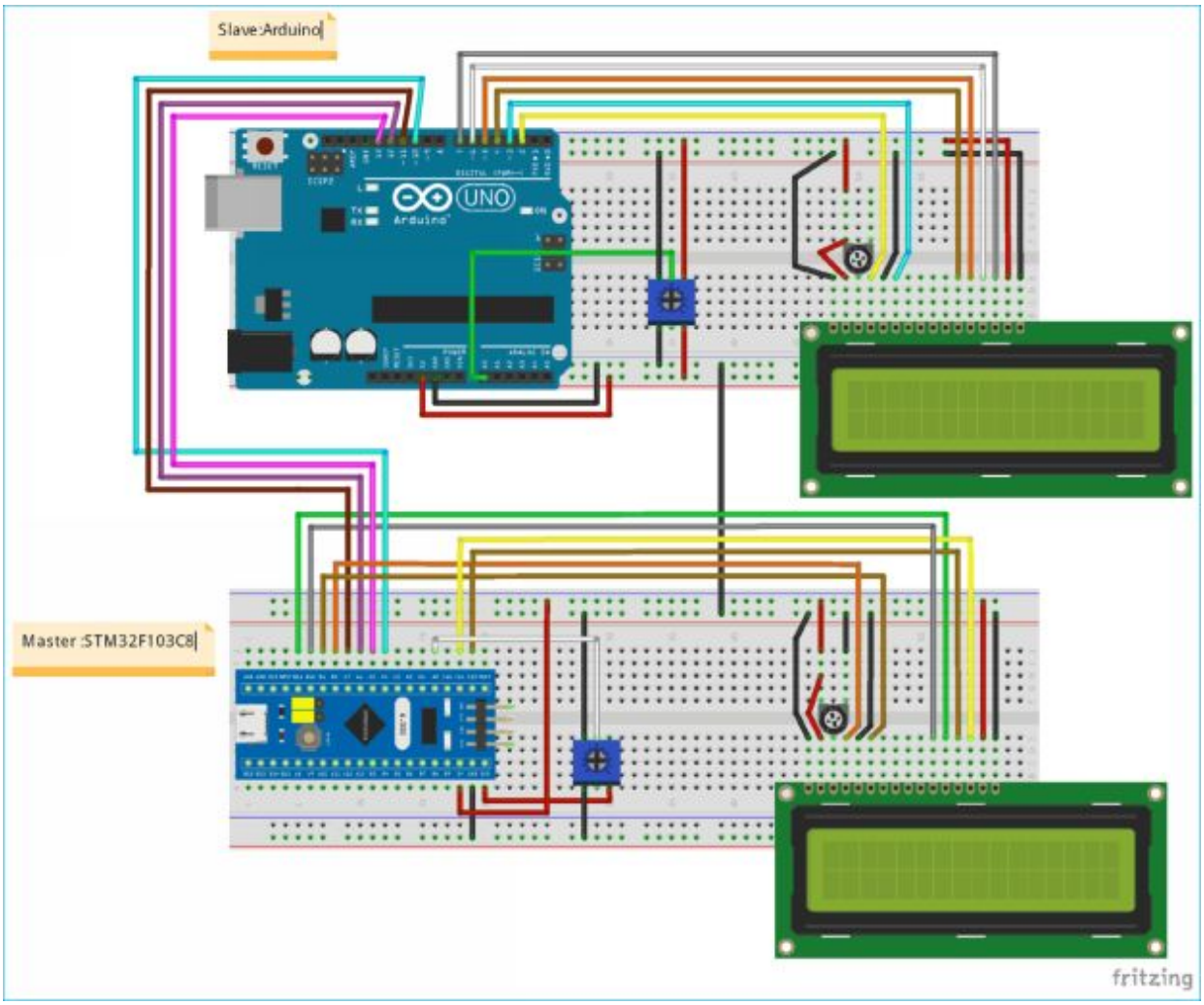
SPI Line	Pin in Arduino
MOSI	11 or ICSP-4
MISO	12 or ICSP-1
SCK	13 or ICSP-3
SS	10

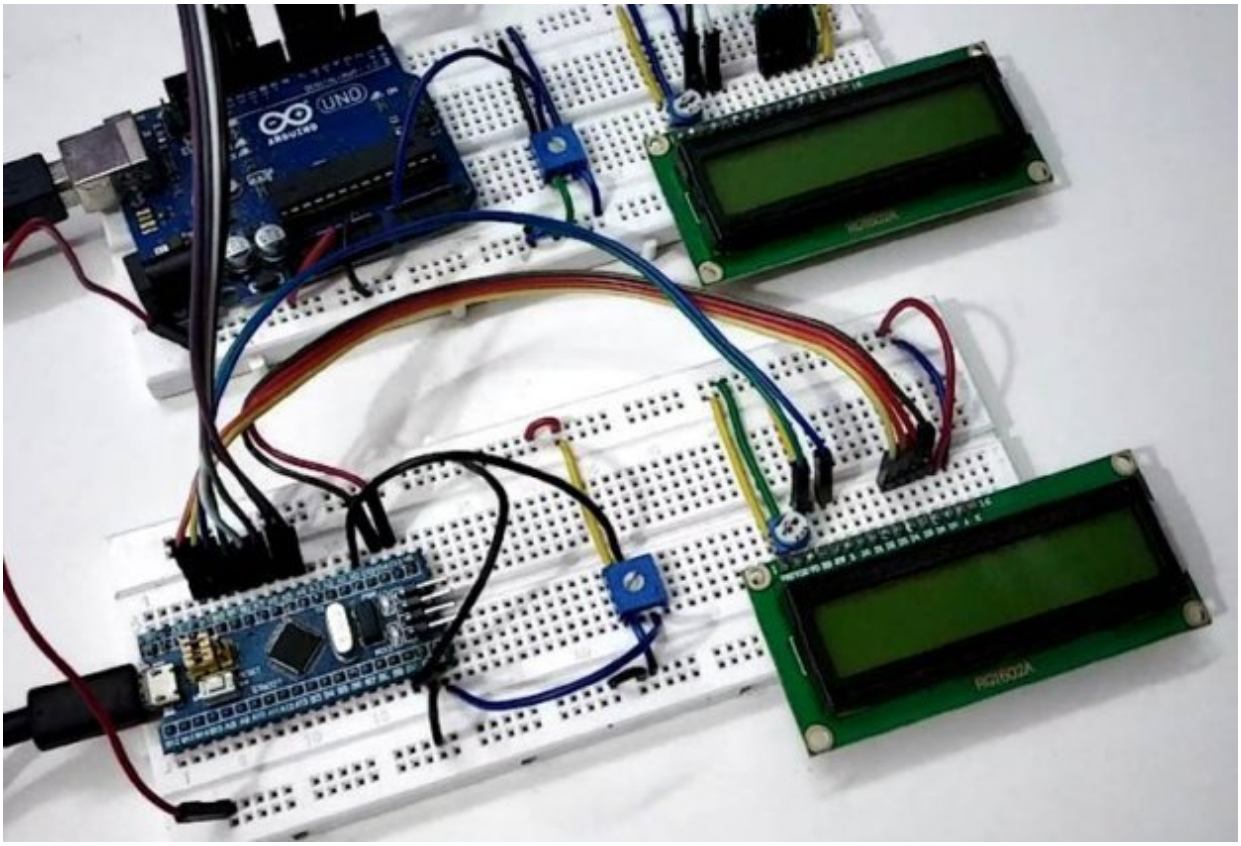
Segments Required

- STM32F103C8
- Arduino
- LCD 16x2 - 2

- 10k Potentiometer - 4
- Breadboard
- Associating Wires

Schematic Diagram along with Links





The Table Below shows the Pins Connected for STM32 SPI correspondence with Arduino.

SPI Pin	STM32F103C8	Arduino
MOSI	PA7	11
MISO	PA6	12
SCK	PA5	13
SS1	PA4	10

The table underneath shows the pins associated for Two LCD (16x2) with STM32F103C8 and Arduino independently.

LCD pin	STM32F103C8	Arduino
VSS	GND	GND
VDD	+5V	+5V

V0	To Potentiometer Centre PIN for LCD contrast	To Potentiometer Centre PIN for LCD contrast
RS	PB0	2
RW	GND	GND
E	PB1	3
D4	PB10	4
D5	PB11	5
D6	PC13	6
D7	PC14	7
A	+5V	+5V
K	GND	GND

Significant:

- Remember to interface the Arduino GND and STM32F103C8 GND together.

STM32 SPI Programming

The writing computer programs is like the Arduino code. The equivalent <SPI.h> library is utilized in programming STM32F103C8. It tends to be modified utilizing USB port without utilizing FTDI software engineer, to study programming STM32 with Arduino IDE follow the connection.

In this STM32 SPI Example, we will utilize Arduino UNO as Slave and STM32F103C8 as Master with Two 16X2 LCD show joined to one another independently. Two Potentiometers are likewise associated with STM32 (PA0) and Arduino (A0) to decide the sending esteems (0 to 255) from ace to slave and slave to ace by changing the potentiometer.

Simple info is taken at STM32F10C8 pin PA0 (0 to 3.3V) by pivoting the potentiometer. At that point this information esteem is changed over into Analog to Digital worth (0 to 4096) and this computerized esteem is

additionally mapped to (0 to 255) as we can send just 8-piece (byte) information through SPI correspondence immediately.

Thus in Slave side we take simple information esteem at Arduino pin A0 from (0 to 5V) by utilizing potentiometer. Furthermore, again this info esteem is changed over into Analog to Digital worth (0 to 1023) and this advanced worth is additionally mapped to (0 to 255)

This instructional exercise has two projects one for ace STM32 and other for slave Arduino.

Ace STM32 SPI Programming Explanation

1. As a matter of first importance we have to incorporate the SPI library for utilizing SPI correspondence capacities and LCD library for utilizing LCD capacities. Likewise characterize LCD pins for 16x2 LCD. Get familiar with interfacing Liquid Crystal Display with STM32 here.

```
#include<SPI.h>
#include<LiquidCrystal.h>
const int rs = PB0, en = PB1, d4 = PB10 , d5 = PB11 , d6 = PC13, d7 = PC14;
LiquidCrystal lcd(rs,en,d4,d5,d6,d7);
```

2. In void arrangement()

- Start Serial Communication at Baud Rate 9600.

```
Serial.begin(9600);
```

- Next start the SPI correspondence

```
SPI.begin();
```

- At that point set the Clock divider for SPI correspondence. I have set divider 16.

```
SPI.setClockDivider(SPI_CLOCK_DIV16);
```

- Next set the SS pin HIGH since we didn't begin any exchange to slave arduino.

```
digitalWrite(SS,HIGH);
```

3. In void circle()

- Before sending any an incentive to slave we have to LOW the slave select an incentive to start move to slave from ace.

```
digitalWrite(SS, LOW);
```

- Next read the simple incentive from the ace STM32F10C8 POT connected to stick PA0.

```
int pot = analogRead(PA0);
```

At that point convert this incentive as far as one byte (0 to 255).

```
byte MasterSend = map(pot,0,4096,0,255);
```

- Here comes the significant advance, in the accompanying articulation we send the changed over POT esteem put away in Mastersend variable to the slave Arduino, and furthermore get an incentive from slave Arduino and put away that in mastereceive variable.

```
Serial.println("Slave Arduino to Master STM32");
Serial.println(MasterReceive  lcd.setCursor(0,0);

lcd.print("Master: STM32");
lcd.setCursor(0,1);
lcd.print("SalveVal:");
lcd.print(MasterReceive  delay(500);
digitalWrite(SS, HIGH);
```

Next presentation those got qualities from the slave arduino with a postponement of 500 microseconds and afterward constantly get and show the qualities.

Note: We use serial.println() to see the outcome in Serial Motor of Arduino IDE.

Slave Arduino SPI Programming Explanation

1. Same as ace, as a matter of first importance we have to incorporate the SPI library for utilizing I2C correspondence capacities and LCD library for utilizing LCD capacities. Additionally characterize LCD pins for 16x2 LCD.

```
#include<SPI.h>
#include<LiquidCrystal.h>
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);     // Define LCD Module Pins (RS,EN,D4,D5,D6,D7)
```

2. In void arrangement()

- We Start Serial Communication at Baud Rate 9600.

```
Serial.begin(9600);
```

- Underneath articulation sets MISO as OUTPUT (Have to Send information to Master IN).So information is sent by means of

MISO of Slave Arduino.

```
pinMode(MISO,OUTPUT);
```

- Presently Turn on SPI in Slave Mode by utilizing SPI Control Register

```
SPCR |= _BV(SPE);
```

- At that point turn ON hinder for SPI correspondence. In the event that an information is gotten from ace the Interrupt Service Routine is called and the got esteem is taken from SPDR (SPI information Register)

```
SPI.attachInterrupt();
```

- The incentive from ace is taken from SPDR and put away in Slavereceived variable. This happens in following Interrupt Routine capacity.

```
ISR (SPI_STC_vect)
{
Slavereceived = SPDR;
received = true;
}
```

3. Next in void circle ()

- Peruse the simple incentive from the Slave Arduino POT connected to stick A0.

```
int pot = analogRead(A0);
```

- Convert that esteem as far as one byte as 0 to 255.

```
Slavesend = map(pot,0,1023,0,255);
```

- Next significant advance is to send the changed over an incentive to the Master STM32F10C8, so place the incentive in the SPDR register. The SPDR register is utilized to send and get values.

```
SPDR = Slavesend;
```

- At that point show the got esteem (SlaveReceive) from Master STM32F103C8 on LCD with a deferral of 500 microseconds and afterward ceaselessly get and show those worth.

```
lcd.setCursor(0,0);
lcd.print("Slave: Arduino");
lcd.setCursor(0,1);
lcd.print("MasterVal:");
Serial.println("Master STM32 to Slave Arduino");
Serial.println(SlaveReceived);
lcd.print(SlaveReceived);
delay(500);
```

By turning the Potentiometer at one side, you can view the shifting qualities on LCD on another side:

So this is the manner by which SPI correspondence happens in STM32. Presently you can interface any SPI sensor with STM32 board.

The total coding for Master STM32 and Slave Arduino is given beneath.

Code

Master STM32 SPI

```
//SPI Master code for STM32F103C8
//SPI Communication between STM32 & Arduino


#include<SPI.h>                                                    // Including Library
for using SPI Communication

#define SS PA4

#include<LiquidCrystal.h>                                                  // Including
LCD display library
```

```
const int rs = PB0, en = PB1, d4 = PB10 , d5 = PB11 , d6 = PC13, d7 = PC14;   // Declaring pin names and pin numbers of lcd

LiquidCrystal lcd(rs,en,d4,d5,d6,d7);                                   // Setting lcd and its paramaters

void setup (void)

{
  lcd.begin(16,2);                                                  // Setting lcd as 16x2 mode
  lcd.setCursor(0,0);                                               // Setting cursor at first row and first column
  lcd.print("HELLO_WORLD");                                         // Puts Hello world in LCD
  delay(3000);                                                      // Delays for 3 seconds
  lcd.clear();                                                      // Clears lcd display



    Serial.begin(9600);                                             // Starts Serial Communication at Baud Rate 9600
  pinMode(SS,OUTPUT);                                               // Puts SS as Output
  SPI.begin();                                                      // Begins the SPI commnuication
  SPI.setClockDivider(SPI_CLOCK_DIV16);                             // Sets clock for SPI communication at 16 (72/16=4.5Mhz)
   digitalWrite(SS,HIGH);                                           // Setting SlaveSelect as HIGH (So master doesnt connnect with slave)
}
void loop(void)
{



    byte MasterSend,MasterReceive;
```

```
   int pot = analogRead(PA0);                                    // Analog read
the input pot value at pin PA0


   MasterSend = map(pot,0,4096,0,255);                           // Used to
convert pot value in terms of 0 to 255 from 0 to 4096


   digitalWrite(SS, LOW);                                        // Starts
communication with Slave connected to master


   MasterReceive=SPI.transfer(MasterSend);                       // Send
the mastersend value to slave also receives value from slave
  Serial.println("Slave Arduino to Master STM32");               // Used
in Serial Monitor
  Serial.println(MasterReceive);                                 // Puts value
Received im Serail Monitor
  lcd.setCursor(0,0);
  lcd.print("Master: STM32");
  lcd.setCursor(0,1);
  lcd.print("SalveVal:");
   lcd.print(MasterReceive);                                     // Puts the
received value from slave arduino
  delay(500);
  digitalWrite(SS, HIGH);                                       // Again make SS
line HIGH so that it doesnt communicate with Slave
  lcd.clear();
}
```

Slave Arduino SPI

```
//SPI Slave Code for Arduino
//SPI Communication between STM32F103C8 & Arduino


#include<SPI.h>                          // Including Library for using SPI
Communication
```

```
#include<LiquidCrystal.h>                // Including LCD display library
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);          // Define LCD Module Pins (RS,EN,D4,D5,D6,D7)
volatile boolean received;
volatile byte SlaveReceived,Slavesend;
void setup()

{
  lcd.begin(16,2);                    // Initilize LCD display
  lcd.setCursor(0,0);                  // Sets Cursor at first line of Display
  lcd.print("Hello world");           // Prints Hello world in LCD
  delay(3000);                        // Delay for 3 seconds
  lcd.clear();                       // Clears LCD display



   Serial.begin(9600);                    // Starts Serial Communication at Baud Rate 9600


   pinMode(MISO,OUTPUT);                  // Sets MISO as OUTPUT (Have to Send data to Master IN (STM32F103C8)
  SPCR |= _BV(SPE);                   // Turn on SPI in Slave Mode
  received = false;
   SPI.attachInterrupt();                        // Interuupt ON is set for SPI commnucation
}

ISR (SPI_STC_vect)                    // Inerrrput routine function
{
  SlaveReceived = SPDR;                        // Value received from master STM32F103C8 is stored in variable slavereceived
  received = true;                    // Sets received as True
}

void loop()
{
  int pot = analogRead(A0);             // Analog read the input pot value from
```

```
analog pin A0
  Slavesend = map(pot,0,1023,0,255);       // Converts the value pot (0-1023) to (0-255) for sending to master stm32



   SPDR = Slavesend;                       // Sends the salvesend value to master STM32F103C8 via SPDR
  lcd.setCursor(0,0);
  lcd.print("Slave: Arduino");
  lcd.setCursor(0,1);
  lcd.print("MasterVal:");
  Serial.println("Master STM32 to Slave Arduino");
  Serial.println(SlaveReceived);                  // Puts the received value from Master STM32F103C8 at Serial Monitor
  lcd.print(SlaveReceived);                       // Puts the received value from Master STM32F103C8 at LCD display
  delay(500);
  lcd.clear();
}
```

2. Programming STM32F103C8 utilizing Keil uVision and STM32CubeMX

STM32 Microcontrollers which uses ARM Cortex M engineering is currently getting famous and are utilized in numerous applications due to its element, cost and execution. We have customized STM32F103C8 utilizing the Arduino IDE in our past instructional exercises. Programming STM32 with Arduino IDE is basic, as there are bunches of libraries accessible for different sensors to play out any undertaking, we simply need to include those libraries in the program. This is a simple method and you may not get into profound finding out about the ARM processors. So now we are getting into the following degree of programming called ARM programming. By this we can, enhance our structure of the code as well as spare memory space by not utilizing pointless libraries.

STMicroelectronics presented a device called STM32Cube MX, which produces essential code as indicated by the peripherals and the chose STM32 board. So we don't have to stress over coding for fundamental drivers and peripherals. Further this created code can be used in Keil uVision for altering as indicated by prerequisite. Lastly the code is scorched into STM32 utilizing ST-Link developer from STMicroelectronics.

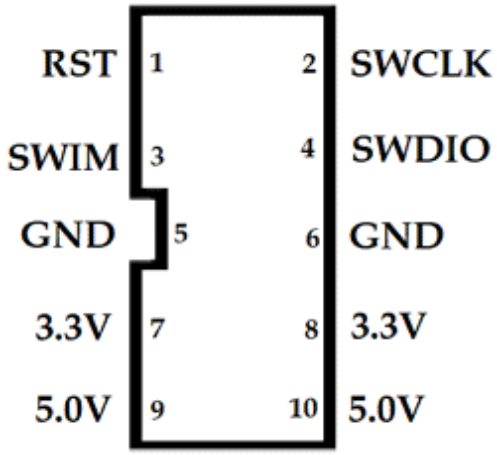
In this instructional exercise we will figure out how to program STM32F103C8 utilizing Keil uVision and STM32CubeMX by doing a straightforward task of interfacing a press catch and LED with the STM32F103C8 Blue Pill board. We will produce the code utilizing STM32Cube MX at that point alter and transfer the code to STM32F103C8 utilizing Keil uVision. Before diving into detail, we will initially find out about ST-LINK developer and STM32CubeMX programming apparatus.

ST-LINK V2

The ST-LINK/V2 is an in-circuit debugger along with software engineer for the STM8 along with STM32 microcontroller families. We can transfer code to STM32F103C8 and other STM8 along with STM32 microcontrollers utilizing this ST-LINK. The single wire interface module (SWIM) and JTAG/sequential wire investigating (SWD) interfaces are utilized to speak with any STM8 or STM32 microcontroller situated on an application board. As STM32 applications utilize the USB max throttle interface to speak with Atollic, IAR, Keil or TASKING coordinated advancement situations, so we can utilize this equipment to program the STM 8 and STM32 microcontrollers.

Above is the picture of ST-LINK V2 dongle from STMicroelectronics that underpins the full scope of STM32 SWD investigating interface, a basic 4-wire interface (counting power), quick and stable. It is affordable in an assortment of hues. The body is made from aluminum amalgam. It has a blue LED sign as it is utilized to watch the functioning situation of the ST-LINK. The pin names are obviously set apart on the shell as should be

obvious in the above picture. It very well may be interfaced with the Keil programming where the program can be flashed to the STM32 microcontrollers. So we should find in this instructional exercise how this ST-LINK developer can be utilized to program STM32 microcontroller. Underneath picture shows the pins of the ST-LINK V2 module.



Note: When associating ST-Link with the PC for first time .We need gadget driver to be introduced. Gadget drivers found in this connection as per your working framework.

STM32CubeMX

STM32CubeMX device is a piece of STMicroelectronics STMCube .This product apparatus puts forth the improvement simple by diminishing advancement attempt, time and cost. STM32Cube incorporates STM32CubeMX which is a graphical programming design device that permits the age of C instatement code utilizing graphical wizards. That code can be utilized in different advancement situations like keil uVision, GCC, IAR and so on. You can download this instrument from the accompanying connection.

STM32CubeMX has following highlights

- Pin out-clash solver

- A clock-tree setting partner
- A force utilization adding machine
- An utility performing MCU fringe setup like GPIO pins, USART and so forth
- An utility performing MCU fringe arrangement for middleware stacks like USB, TCP/IP and so on

Materials Required

- Equipment
- STM32F103C8 Blue Pill board
- ST-LINK V2
- Press Button
- Driven

- Breadboard

- Jumper Wires

Programming

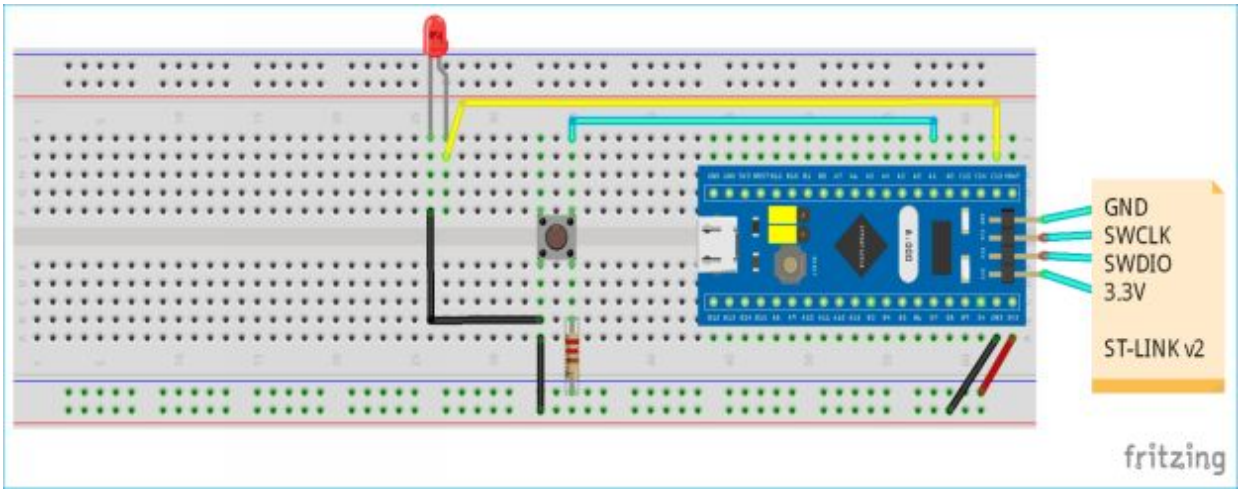
- STM32CubeMX Code Generation Tool (Link)

- Keil uVision 5(link)

- Drivers for ST-Link V2 (interface)

Circuit Diagram and Connections

The following is the circuit chart to just interface a LED with STM32 board utilizing a pushbutton.



Association between ST-LINK V2 and STM32F103C8

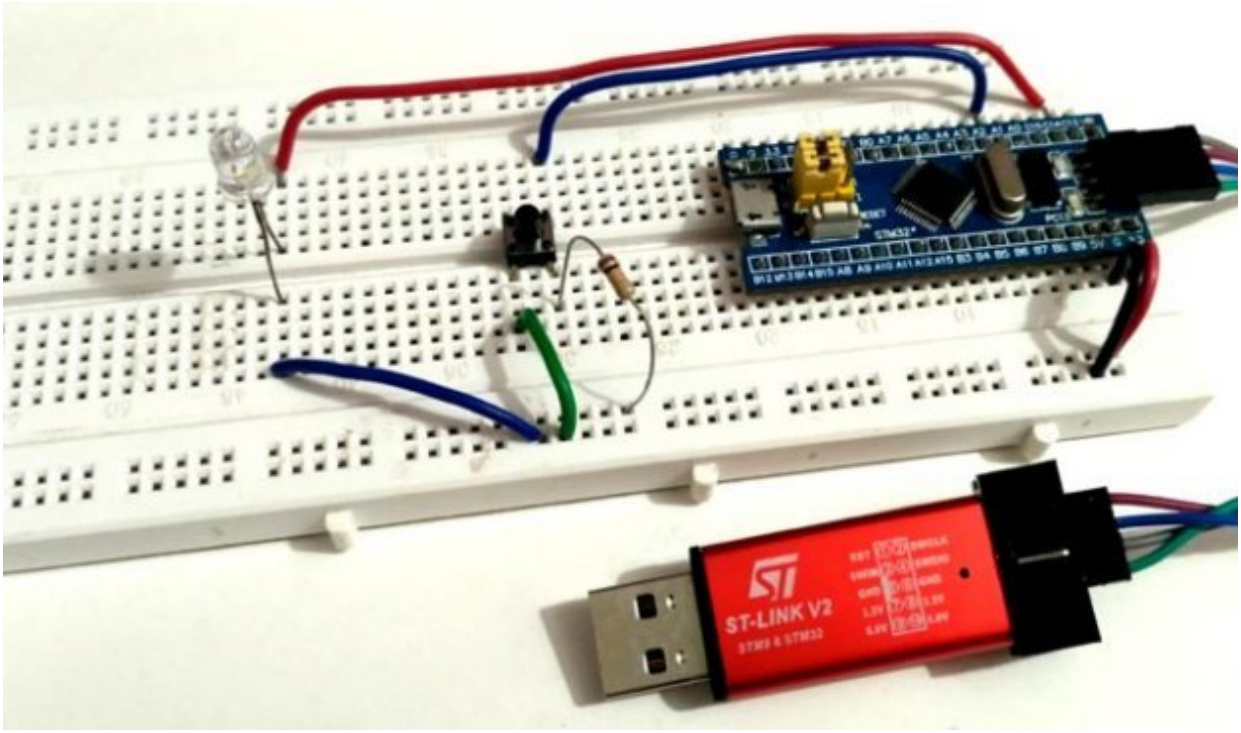
Here the STM32 Blue Pill board is fueled from the ST-LINK which is associated with the PC's USB port. So we require not to control the STM32 independently. The table underneath shows the association between ST-Link and Blue pill board.

STM32F103C8	ST-Link V2
GND	GND
SWCLK	SWCLK
SWDIO	SWDIO
3V3	3.3V

Driven and Push Button

The LED is utilized to show the yield from Blue Pill board when a press button is squeezed. Driven's anode is associated with the pin PC13 of the Blue Pill board and cathode is grounded.

A press button is associated with give contribution to the pin PA1 of Blue Pill board. We should likewise go through a force resistor of significant worth 10k on the grounds that the pin may coast with no info when the catch is discharged. One finish of the press button is associated with ground and opposite end to stick PA1 and a draw up resistor of 10k is additionally associated with 3.3V of Blue Pill board.

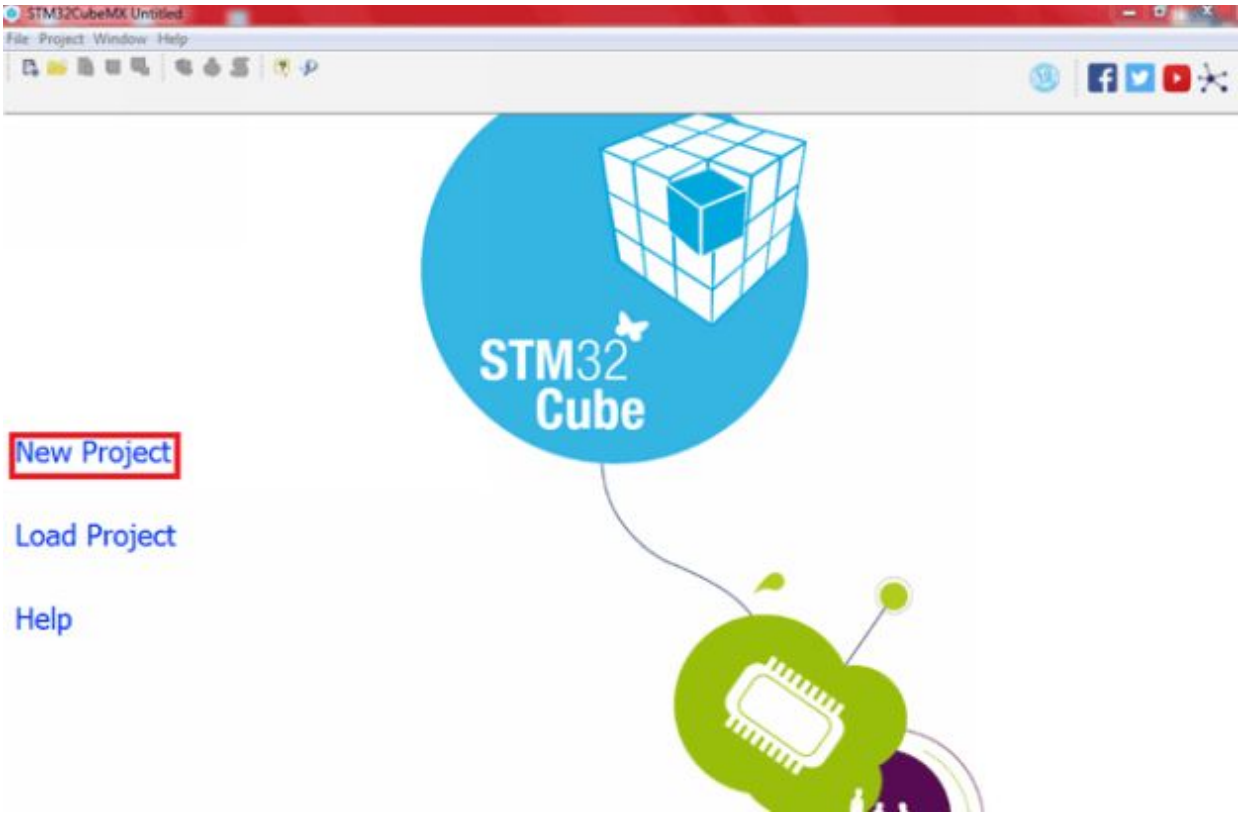


Making and consuming a program into STM32 utilizing Keil uVision along with ST-Link

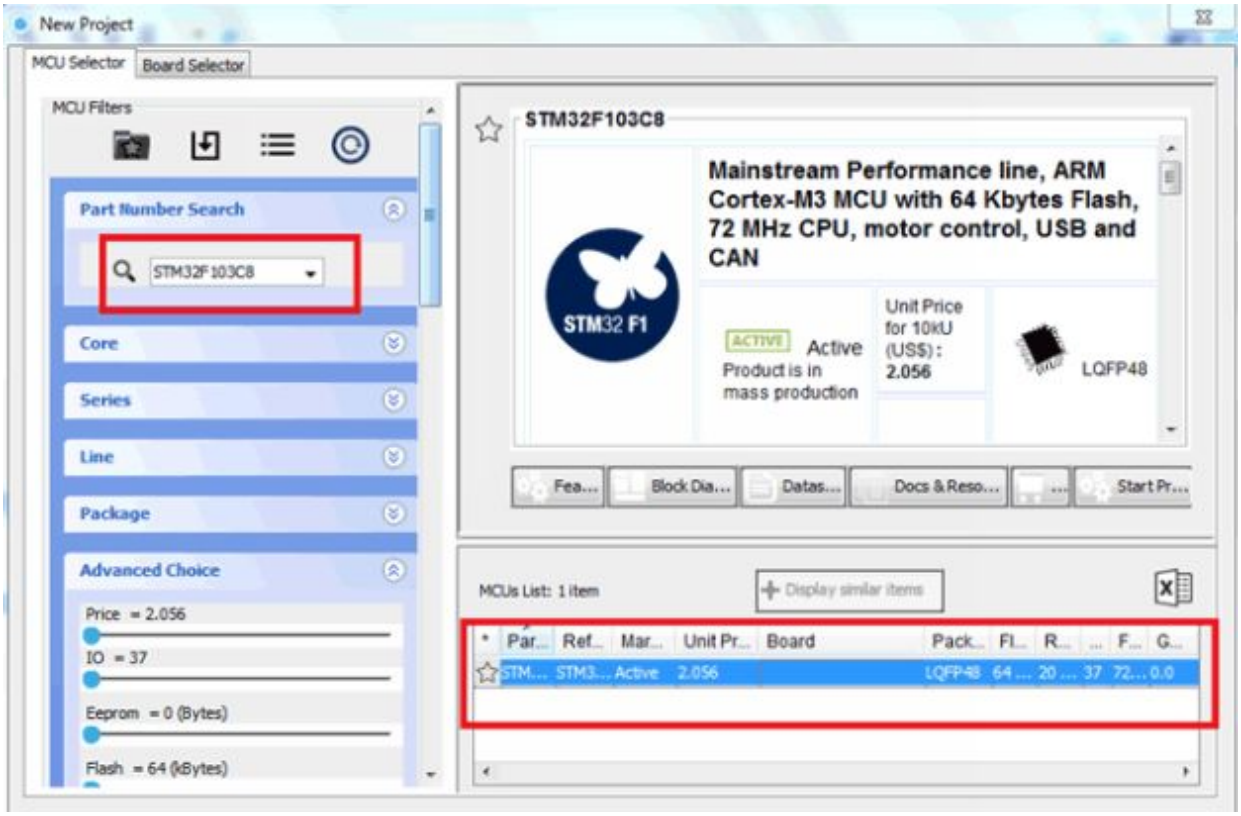
Stage 1:- First introduce all the gadget drivers for ST-LINK V2, programming instruments STM32Cube MX and Keil uVision and introduce important bundles for STM32F103C8.

Stage 2:- 2nd step is Open >> STM32Cube MX

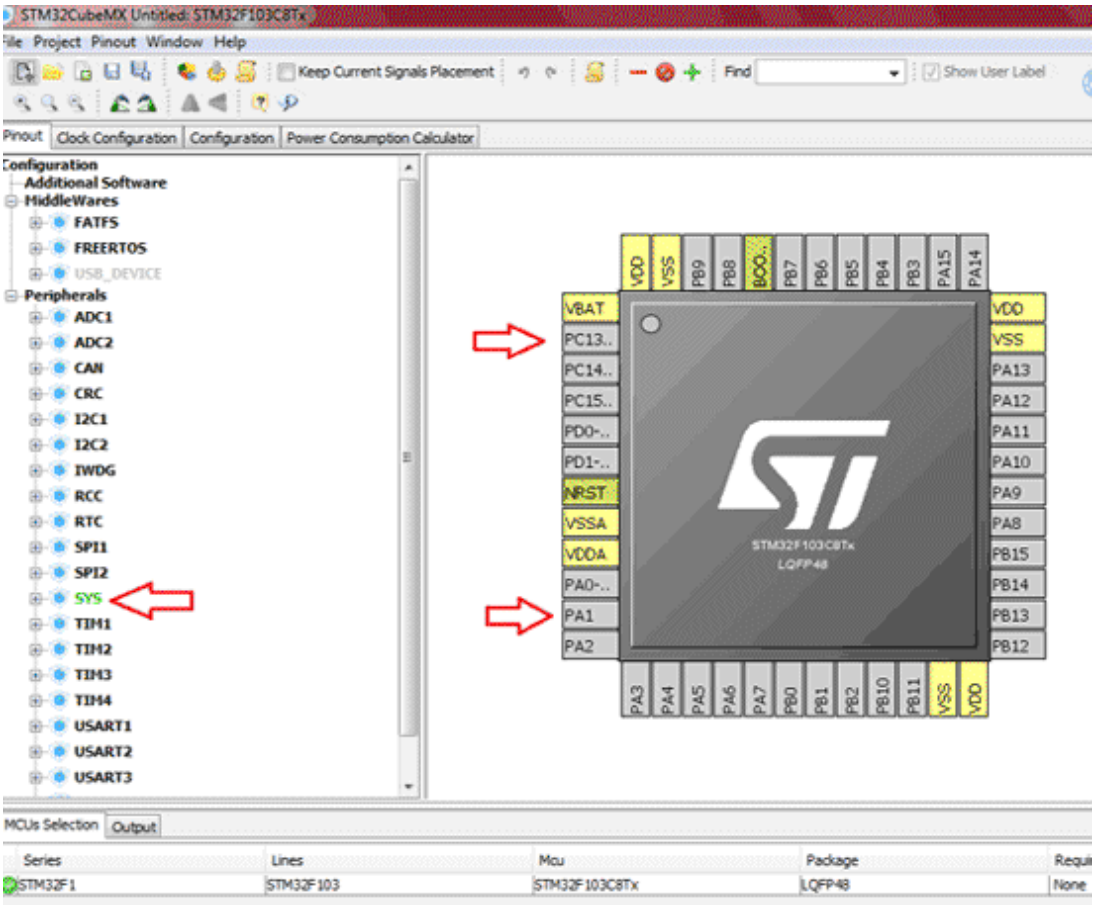
Stage 3:- Then Click on New Project



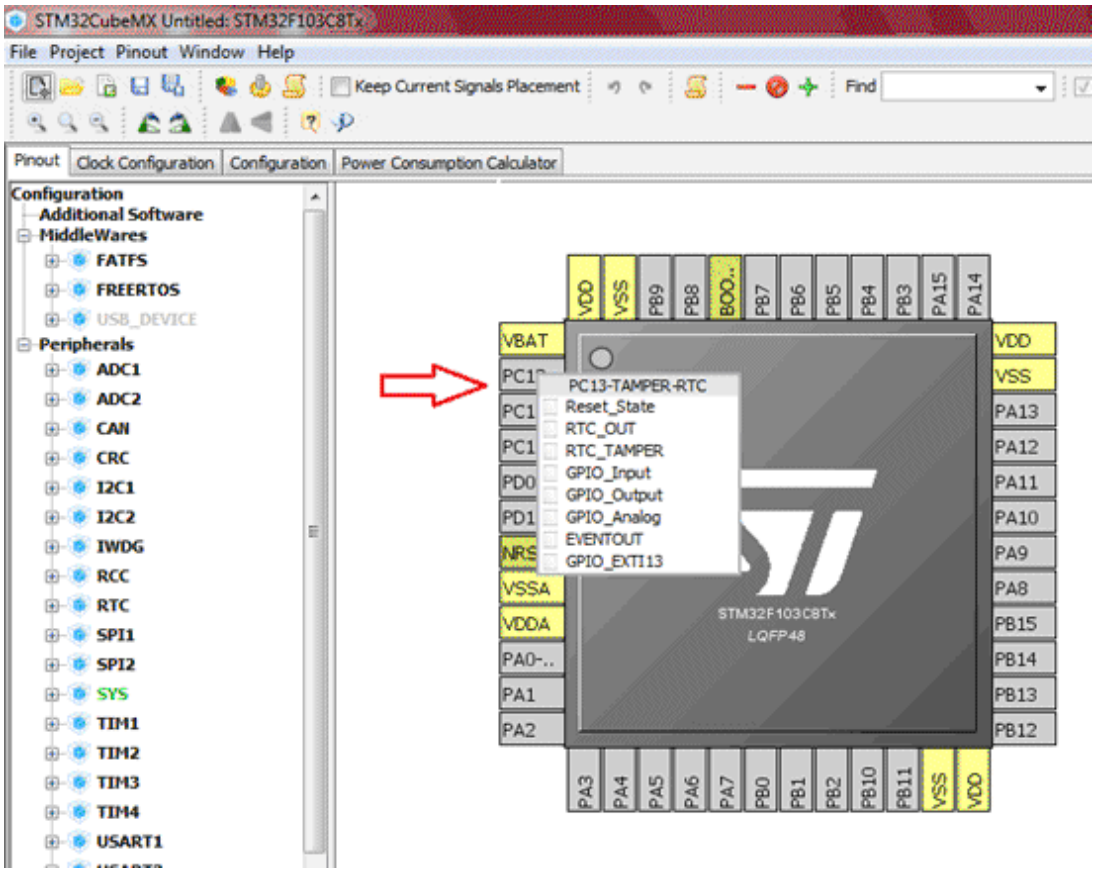
Stage 4:- After that search and select our microcontroller STM32F103C8



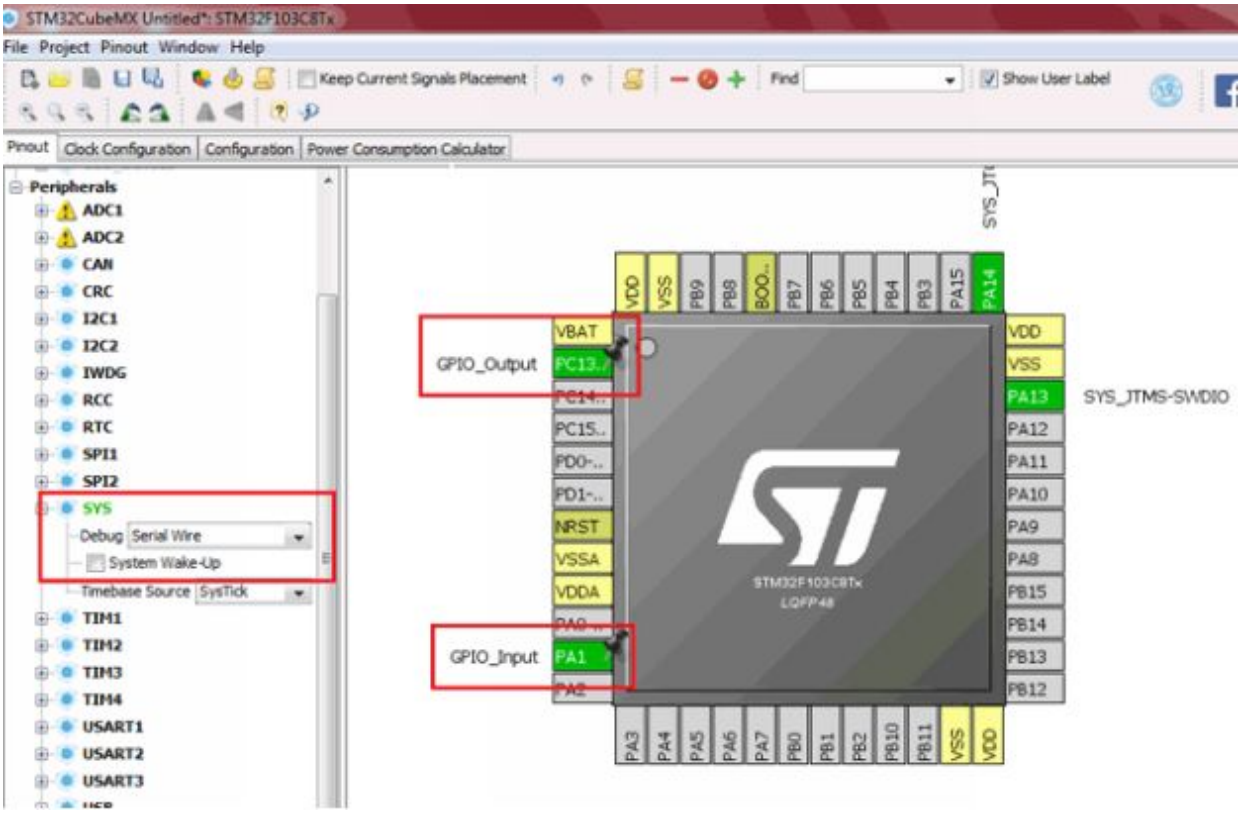
Stage 5:- Now the pin-out sketch of STM32F103C8 shows up, here we can set the pin designs. We can likewise choose our pins in the peripherals segment as per our venture.



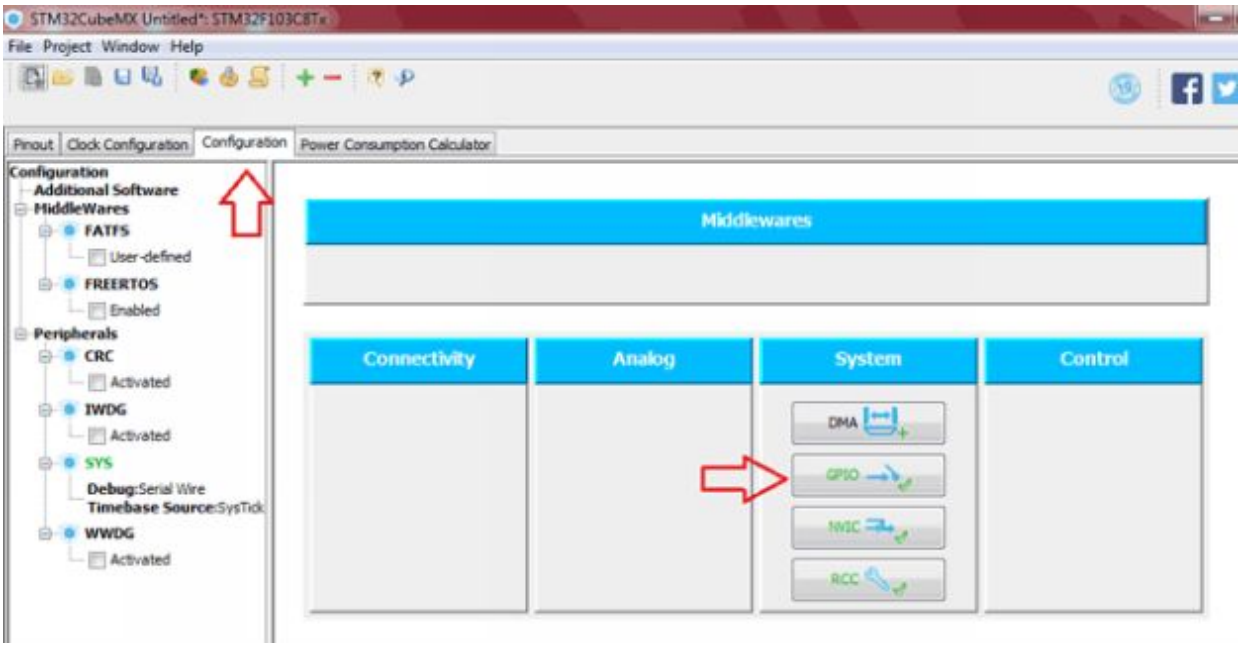
Stage 6:- You can similarly tap on the pin straightforwardly and a rundown shows up, presently select the necessary pin setup.



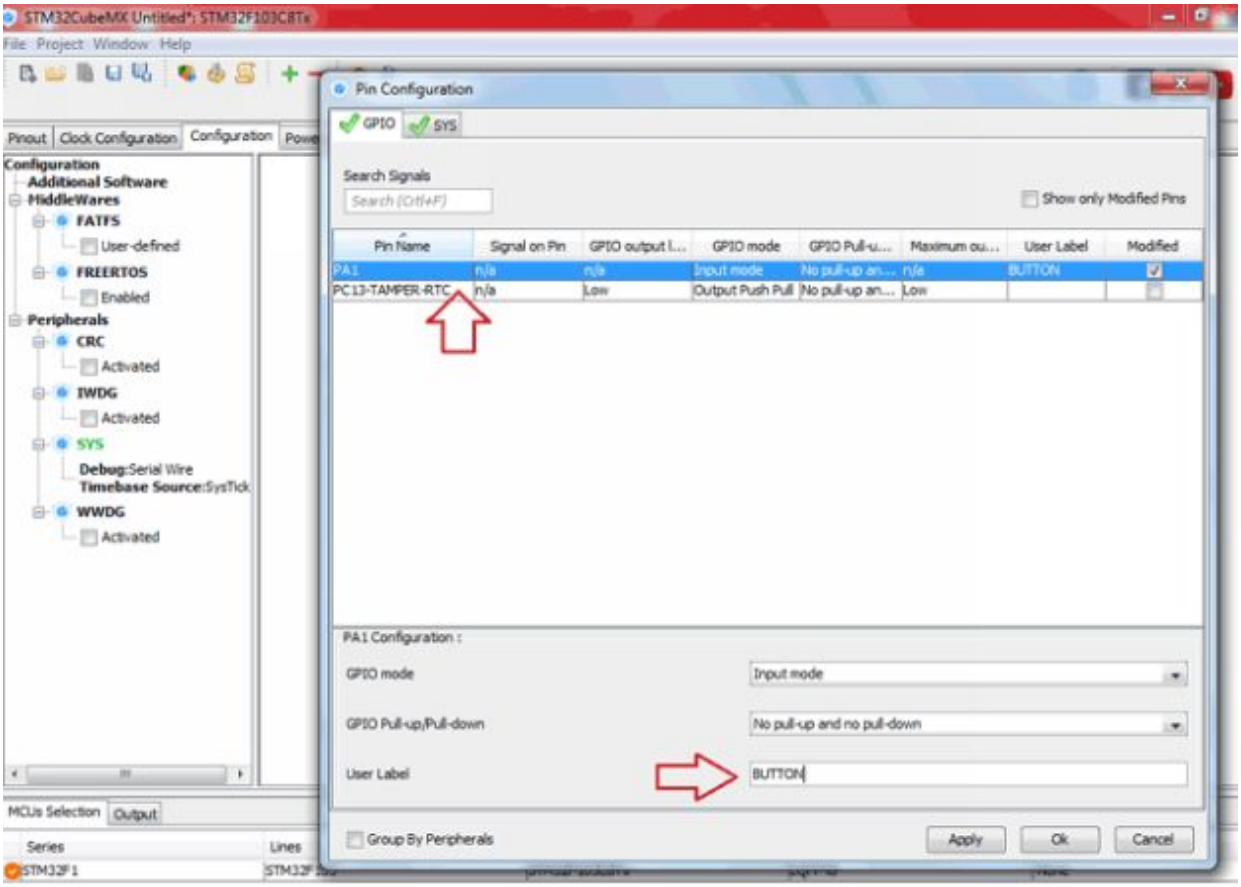
Stage 7:- For our undertaking we have chosen PA1 as GPIO INPUT, PC13 as GPIO OUTPUT and SYS investigate as SERIAL WIRE, here just we associate the ST-LINK SWCLK and SWDIO pins. The chose and arranged pins show up in GREEN shading. You can take note of that in underneath picture.



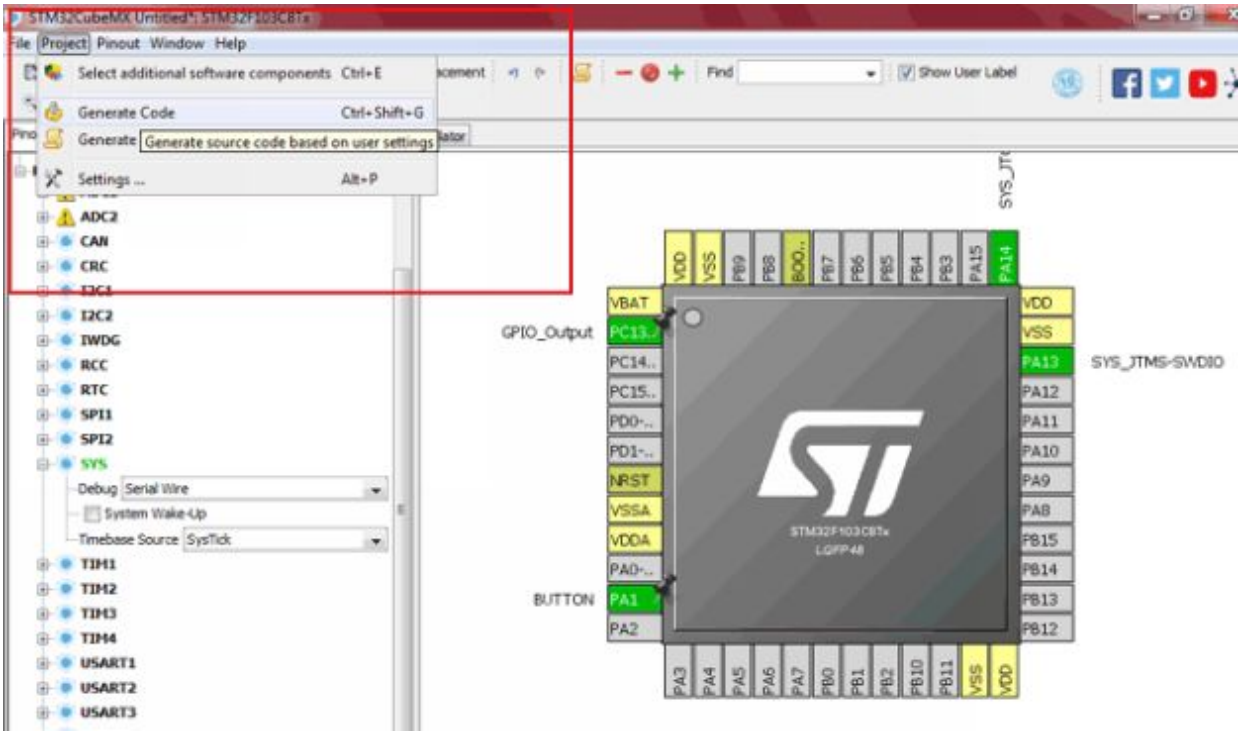
Stage 8:- Next under the Configuration tab, select GPIO to set GPIO pin arrangements for the pins we have chosen.



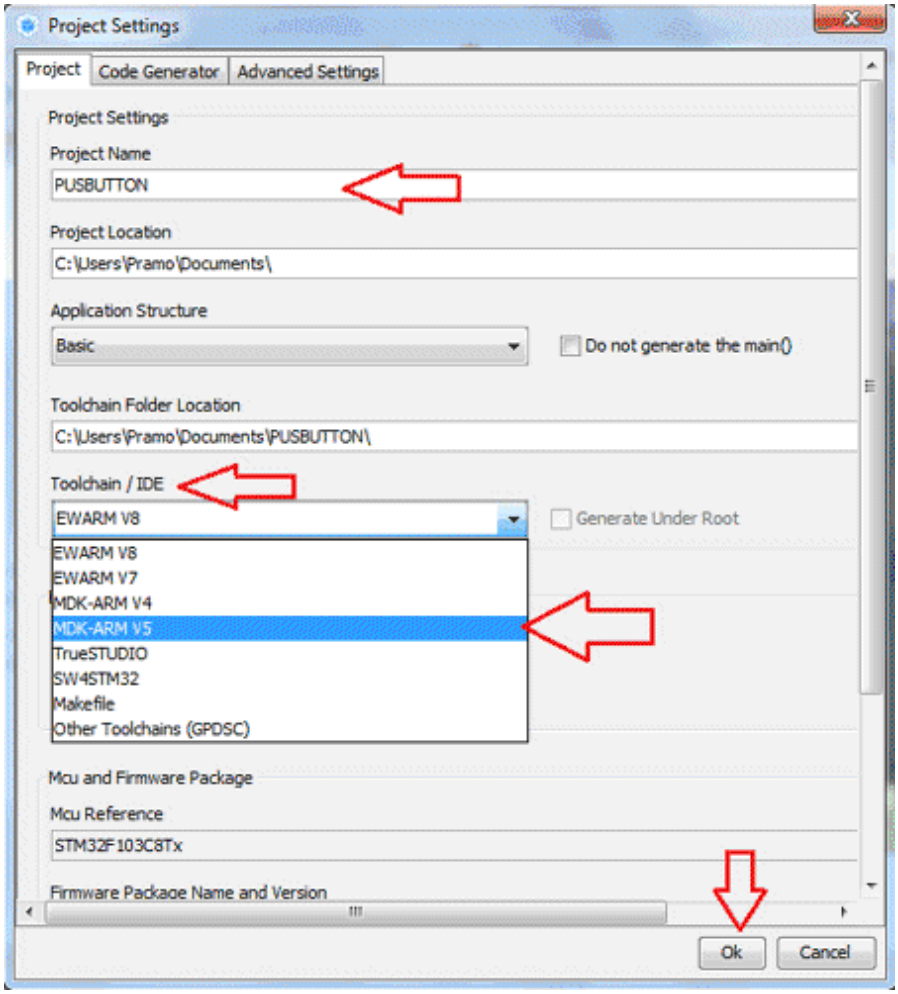
Stage 9:- Next in this pin arrangement box we can design User Label for pins we are utilizing, that is client characterized pin names.



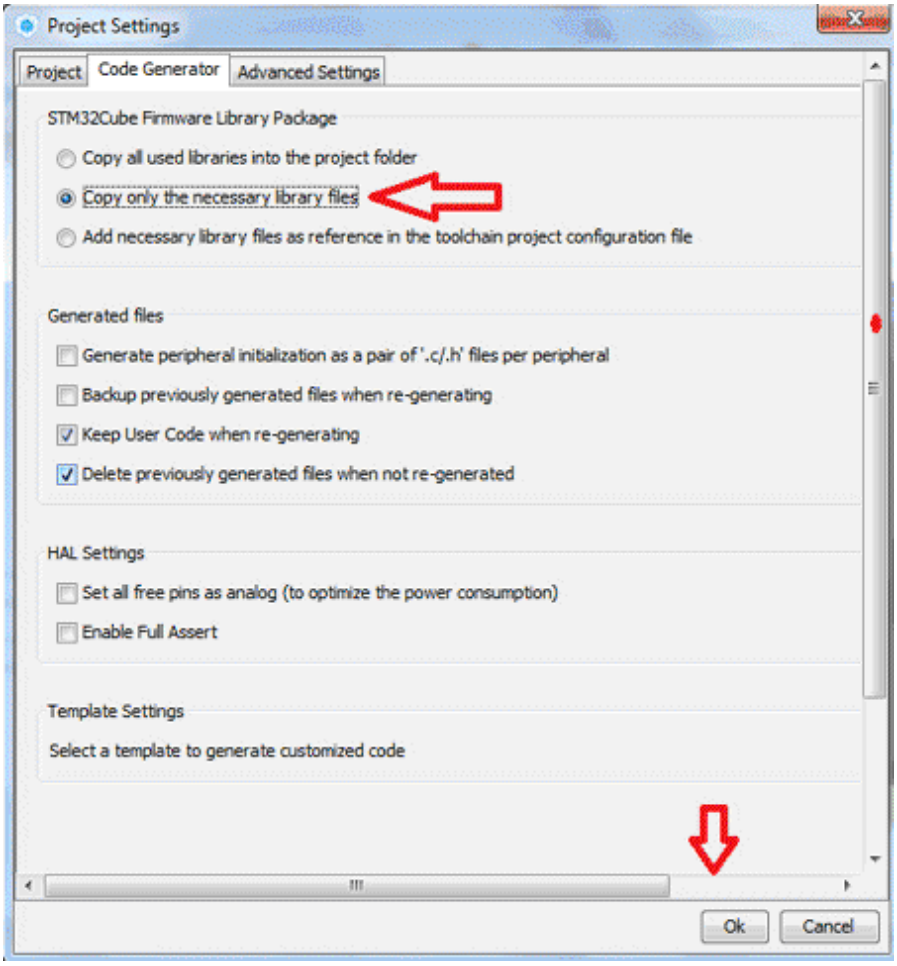
Stage 10:- After that click on Project >> Generate Code.



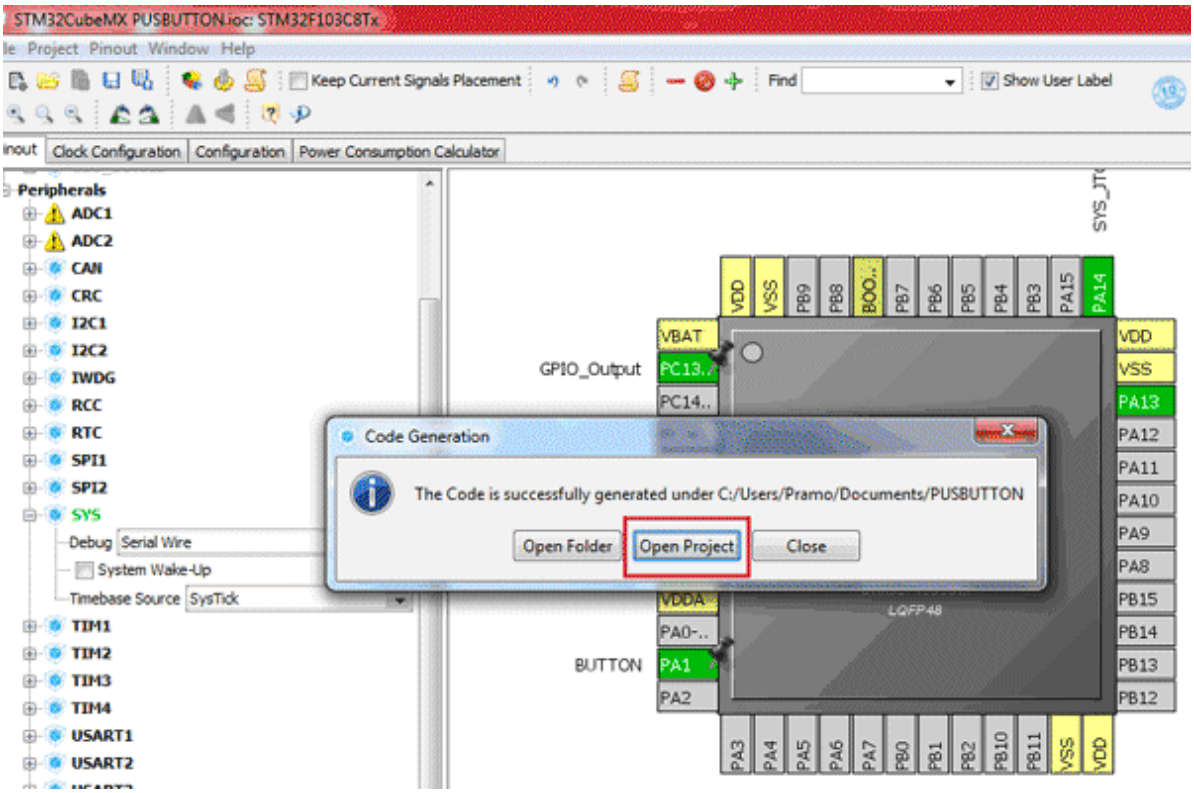
Stage 11:- Now the task settings discourse box shows up. In this crate pick your venture name and area and select the advancement condition .We are utilizing Keil so select MDK-ARMv5 as IDE.



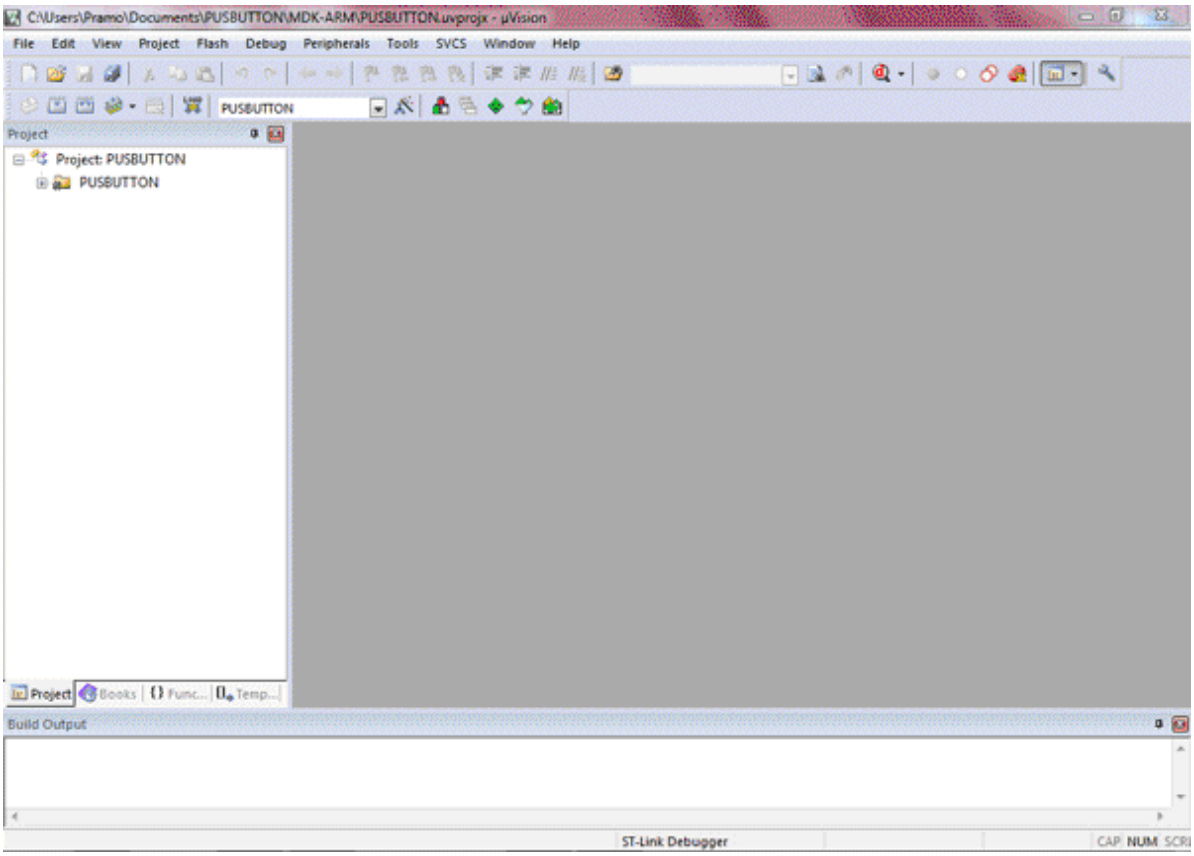
Stage 12:- Next under Code Generator tab, select Copy just the important library records and afterward click OK.



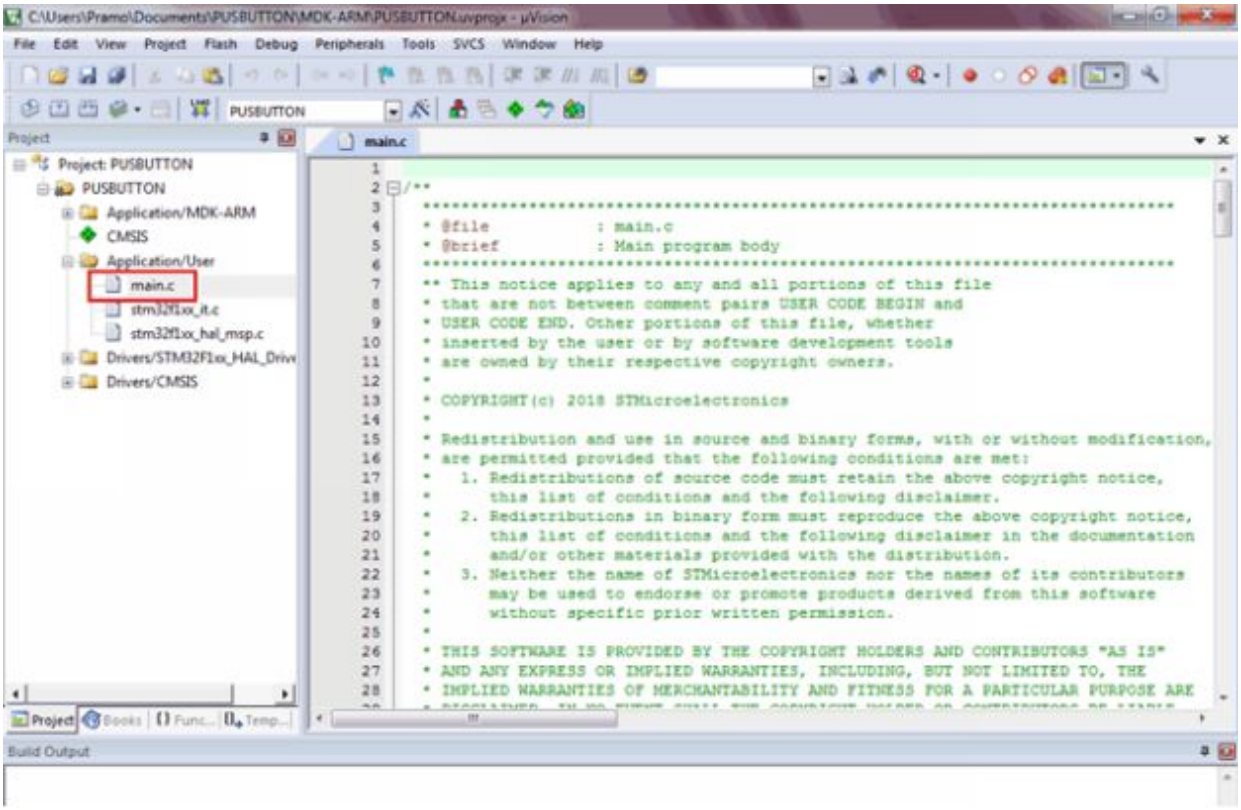
Stage 13:- Now the code age discourse box shows up. Select Open Project to open undertaking consequently the created code in Keil uvsion.



Stage 14:- Now the Keil uVision device opens with our created code in STM32CubeMx with a similar task name with fundamental library and codes that are designed for the pins we chose.

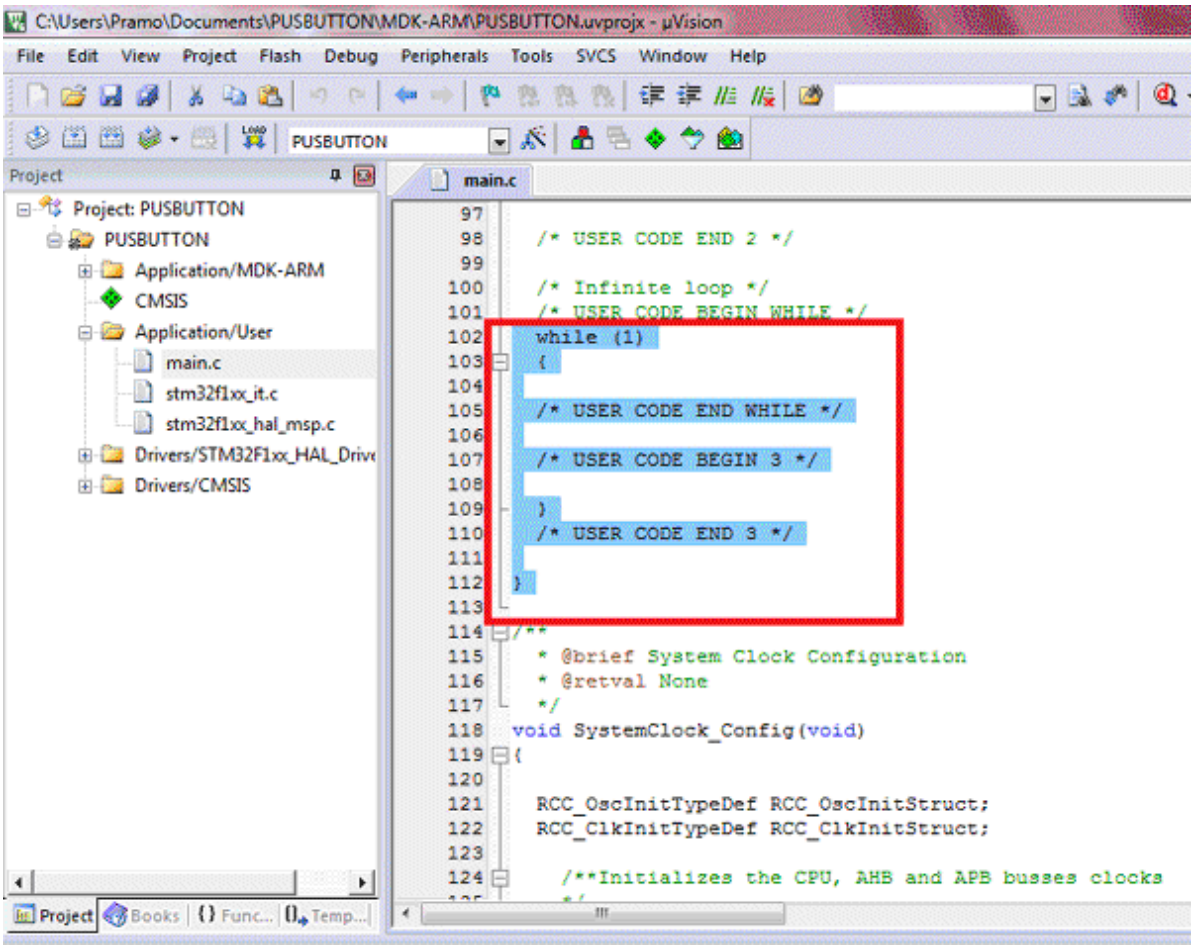


Stage 15:- Now we simply need to incorporate the rationale to play out some activity at the yield LED (pin PC13) when the catch is squeezed and discharged at the General Purpose Input/Output input (pin PA1). So select our main.c program to incorporate a few codes.

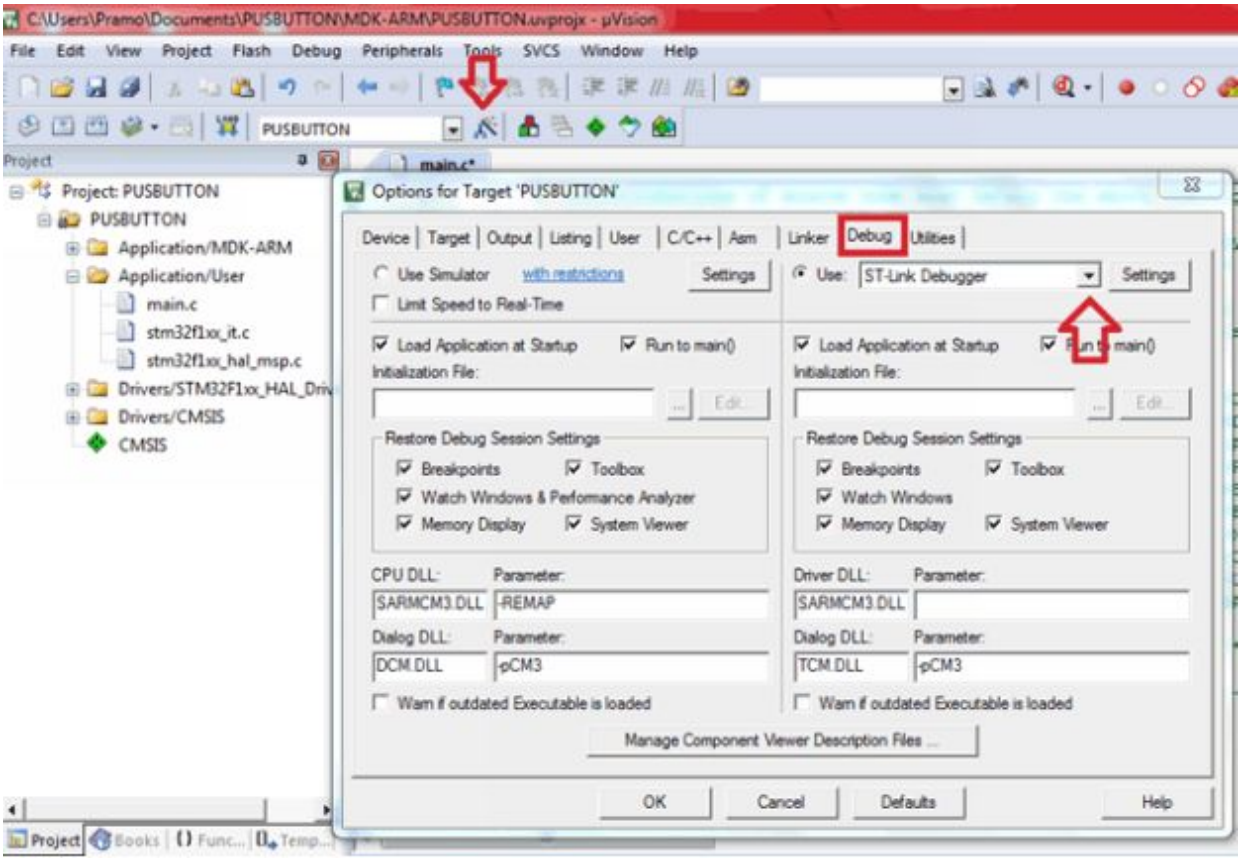


Stage 16:- Now include the code in the while(1) circle, see the underneath picture where I featured that segment to run the code consistently.

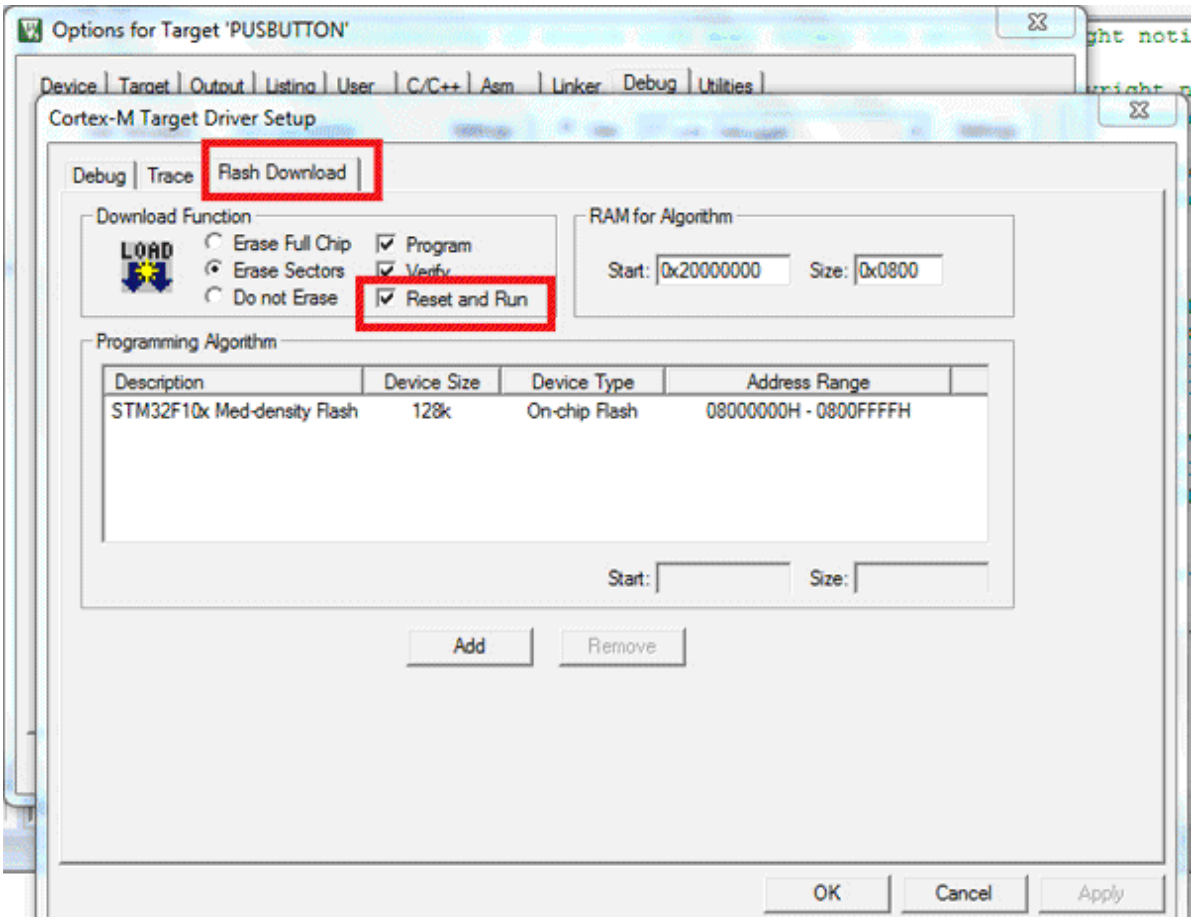
```
    while (1)
    {
  if(HAL_GPIO_ReadPin(BUTN_GPIO_Port,BUTN_Pin)==0)        //=>
DETECTS Button is Pressed
    {
    HAL_GPIO_WritePin(LEDOUT_GPIO_Port,LEDOUT_Pin,1);
//To make output high when button pressesd
    }
    Else
    {
    HAL_GPIO_WritePin(LEDOUT_GPIO_Port,LEDOUT_Pin,0);
//To make output Low when button de pressed
    }
    }
```



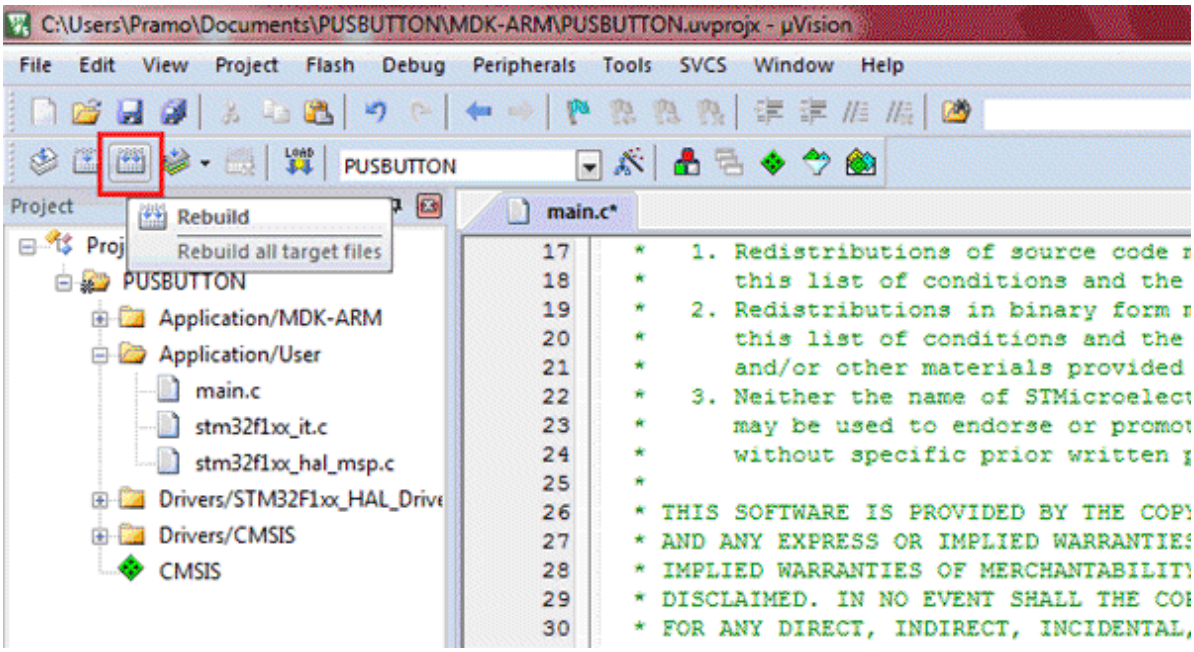
Stage 17:- After wrap up the code, click the Options for Target symbol under the troubleshoot tab select ST-LINK Debugger



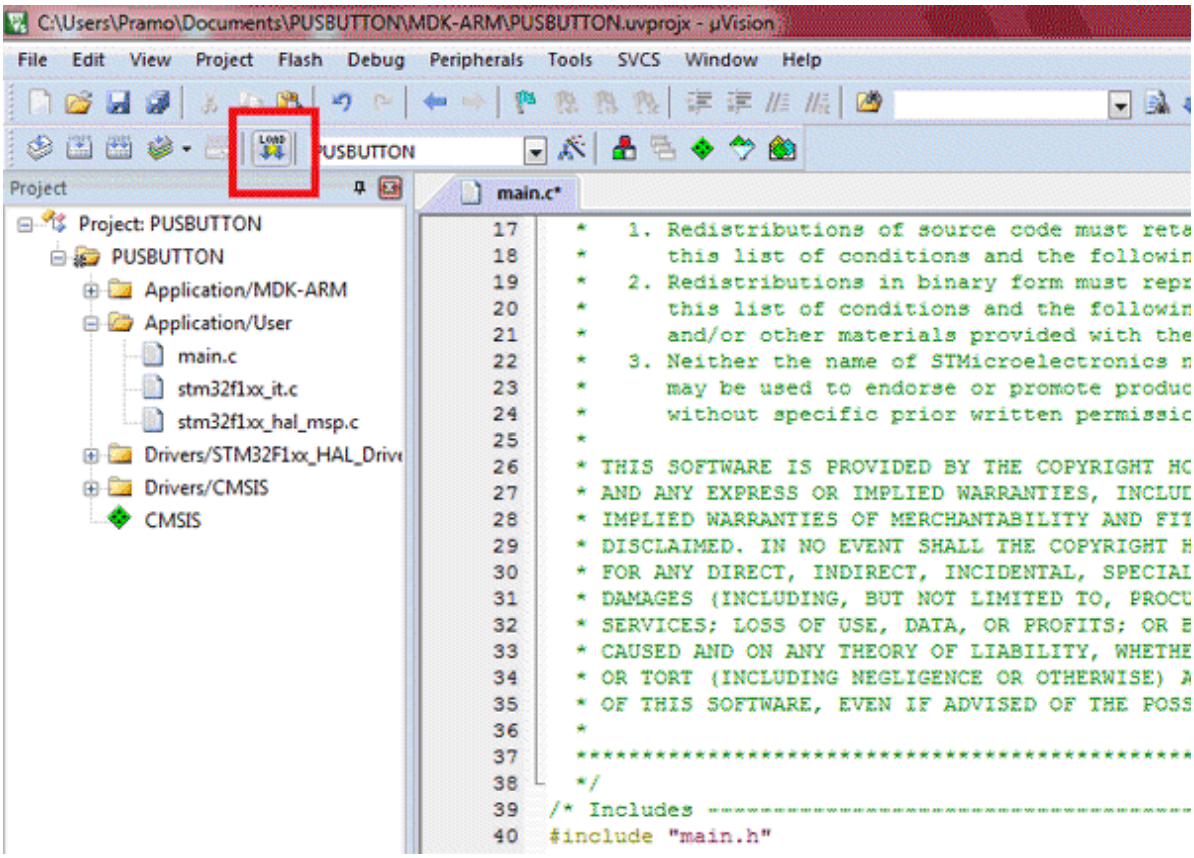
Additionally, click on Settings catch and afterward under Flash Download tab tick the Reset and Run check box and snap 'alright'.



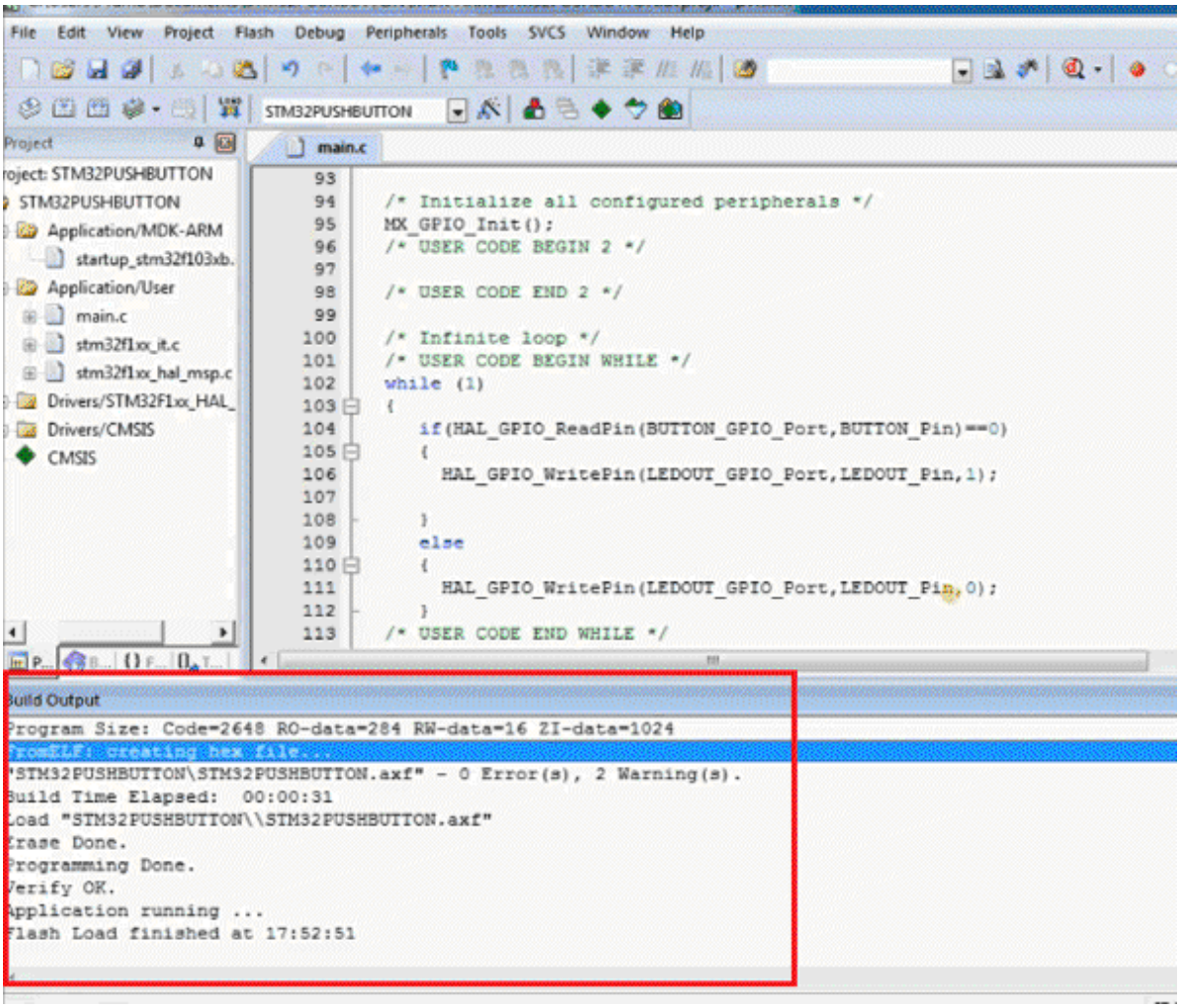
Stage 18:- Now click on Rebuild symbol to revamp all objective records.



Stage 19:- Now you can connect the ST-LINK to PC with the circuit associations prepared and click on the DOWNLOAD symbol or press F8 to streak the STM32F103C8 with the code you have created and altered.

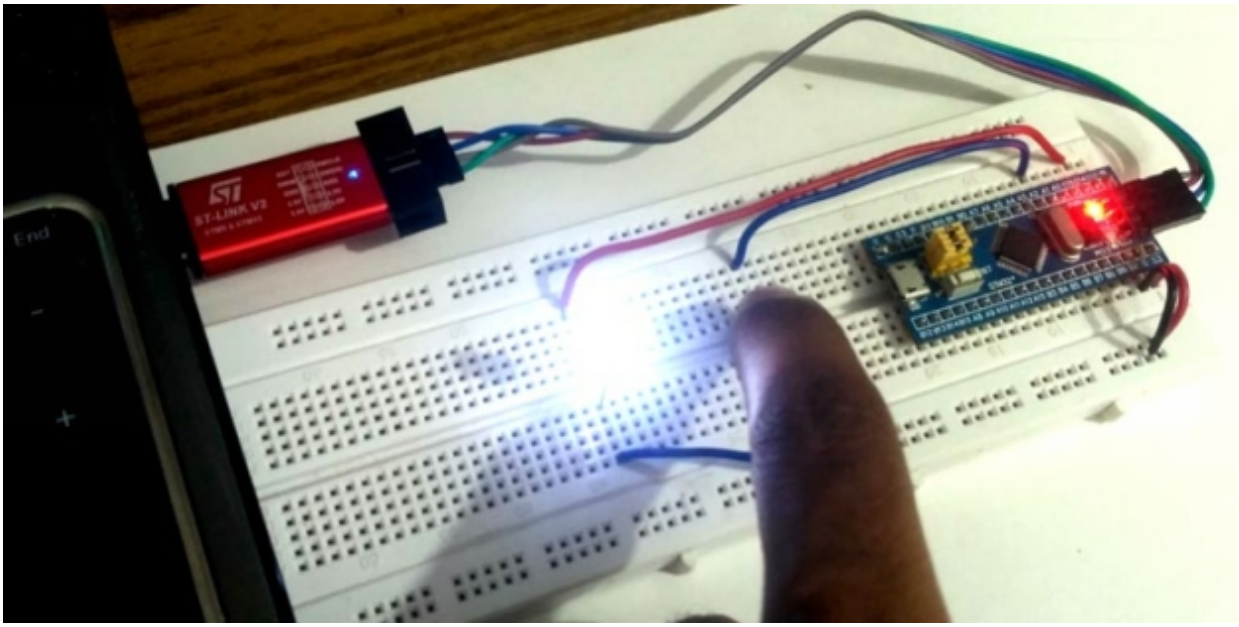


Stage 20:- You can view the glimmering sign at the base of the keil uVision window.



Yield of Keil Programmed STM32 Board

Presently when we press the press button, LED turn On and when we discharge it, the LED kills.



Program

The principle part which we have included the produced program is demonstrated as follows. This underneath code should be remembered for while(1) of main.c program produced by the STM32CubeMX. You can return to Step 15 to stage 17 to find out how it ought to be included main.c program.

```
    while (1)
    {
  if(HAL_GPIO_ReadPin(BUTN_GPIO_Port,BUTN_Pin)==0)        //=>
DETECTS Button is Pressed
    {
    HAL_GPIO_WritePin(LEDOUT_GPIO_Port,LEDOUT_Pin,1);
//To make output high when button pressesd
    }
    Else
    {
    HAL_GPIO_WritePin(LEDOUT_GPIO_Port,LEDOUT_Pin,0);
//To make output Low when button de pressed
    }
    }
```

Complete procedure of making and transferring venture into STM32 board. Likewise the total code of main.c record is given underneath including the above given code.

Further, you can locate our total arrangement of STM32 extends here.

Code

```
/**
 ******************************************************************************
 * @file           : main.c
 * @brief          : Main program body
 ******************************************************************************
 ** This notice applies to any and all portions of this file
 * that are not between comment pairs USER CODE BEGIN and
 * USER CODE END. Other portions of this file, whether
 * inserted by the user or by software development tools
 * are owned by their respective copyright owners.
 *
 * COPYRIGHT(c) 2018 STMicroelectronics
 *
   * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *   1. Redistributions of source code must retain the above copyright notice,
 *      this list of conditions and the following disclaimer.
  *    2. Redistributions in binary form must reproduce the above copyright notice,
   *          this list of conditions and the following disclaimer in the documentation
 *      and/or other materials provided with the distribution.
  *     3. Neither the name of STMicroelectronics nor the names of its contributors
  *         may be used to endorse or promote products derived from this software
 *      without specific prior written permission.
 *
```

```
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
  * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
    * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
      * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *
  ********************************************************************************
  */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f1xx_hal.h"
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
/* USER CODE BEGIN PFP */
```

```
/* Private function prototypes -----------------------------------------*/
/* USER CODE END PFP */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
 *
  * @retval None
 */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
 HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
 SystemClock_Config();
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
 MX_GPIO_Init();
  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
 {
            if(HAL_GPIO_ReadPin(BUTN_GPIO_Port,BUTN_Pin)==0)   //=>
Button is Pressed
{
            HAL_GPIO_WritePin(LEDOUT_GPIO_Port,LEDOUT_Pin,1);
}
else //=>Button is released
{
```

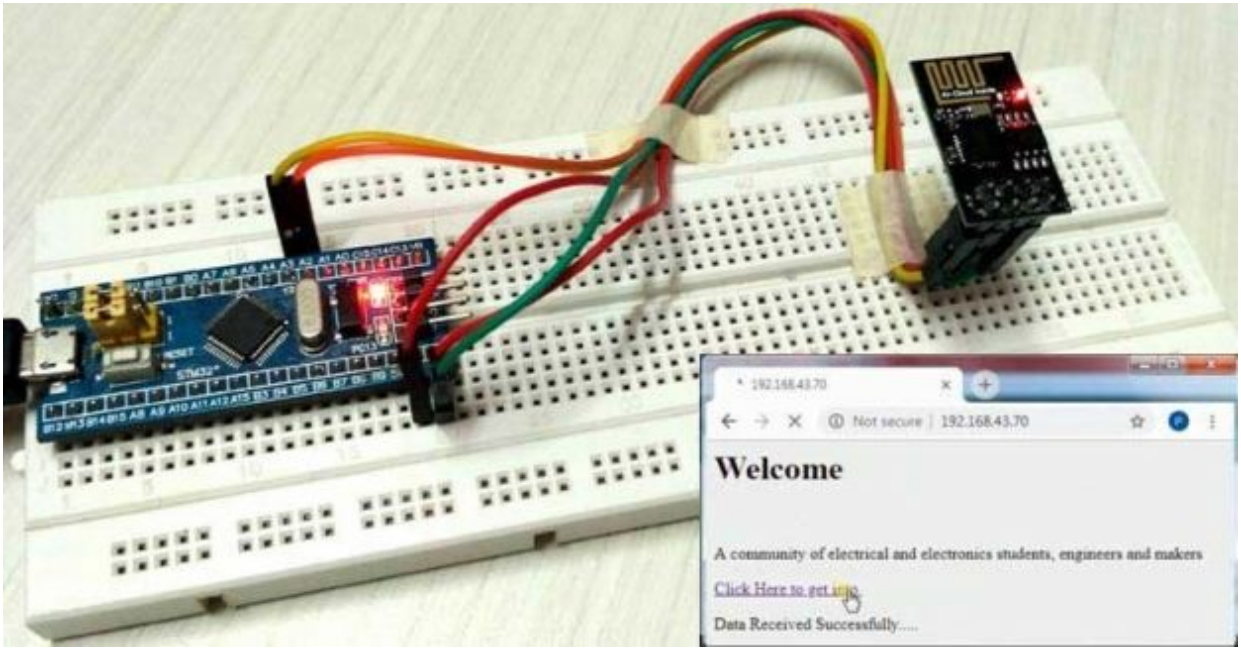
```
            HAL_GPIO_WritePin(LEDOUT_GPIO_Port,LEDOUT_Pin,0);
}
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
 }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
  * @retval None
 */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;
    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = 16;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
    /**Initializes the CPU, AHB and APB busses clocks
    */
                                    RCC_ClkInitStruct.ClockType                =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0)
!= HAL_OK)
```

```
 {
   _Error_Handler(__FILE__, __LINE__);
 }
   /**Configure the Systick interrupt time
   */
 HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);
   /**Configure the Systick
   */
 HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);
  /* SysTick_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}
/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
*/
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  /* GPIO Ports Clock Enable */
 __HAL_RCC_GPIOC_CLK_ENABLE();
 __HAL_RCC_GPIOA_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
              HAL_GPIO_WritePin(LEDOUT_GPIO_Port,       LEDOUT_Pin,
GPIO_PIN_RESET);
  /*Configure GPIO pin : LEDOUT_Pin */
  GPIO_InitStruct.Pin = LEDOUT_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LEDOUT_GPIO_Port, &GPIO_InitStruct);
  /*Configure GPIO pin : BUTN_Pin */
  GPIO_InitStruct.Pin = BUTN_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
```

```
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(BUTN_GPIO_Port, &GPIO_InitStruct);
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @param  file: The file name as string.
  * @param  line: The line in file as a number.
  * @retval None
 */
void _Error_Handler(char *file, int line)
{
  /* USER CODE BEGIN Error_Handler_Debug */
   /* User can add his own implementation to report the HAL error return
state */
 while(1)
 {
if(HAL_GPIO_ReadPin(BUTN_GPIO_Port,BUTN_Pin)==0)               //=>
DETECTS Button is Pressed
    {
         HAL_GPIO_WritePin(LEDOUT_GPIO_Port,LEDOUT_Pin,1);   //To
make output high when button pressesd
    }
    else
    {
         HAL_GPIO_WritePin(LEDOUT_GPIO_Port,LEDOUT_Pin,0);   //To
make output Low when button de pressed
    }
 }
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
```

```
  * @param  line: assert_param error line source number
  * @retval None
 */
void assert_failed(uint8_t* file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
   /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
/**
  * @}
 */
/**
  * @}
 */
/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

3. Interfacing ESP8266 with STM32F103C8: Creating a Webserver



In the event that you consider future advances, at that point the two names, which promptly comes in your brain, are Artificial Intelligence (AI) and Internet of Things (IoT). Simulated intelligence is still in its underlying stage and there is part more to be created. However, IoT is in Growth stage and part of IoT based items are as of now there in the market. Additionally there are numerous instruments and equipment accessible in the market to make your item chatting with 'things' in the web. Among them ESP8266 is the most popular, modest and simple to-utilize module, which can interface your equipment to the Internet.

We have created part of IoT Projects utilizing ESP8266, which not just incorporates essential interfacing with different microcontrollers like Arduino, PIC, AVR yet additionally incorporates numerous canny undertakings like IOT based Air Pollution Monitoring, Vehicle Tracking on Google Maps, IOT based Voice Controlled Home Automation and so on. In this instructional exercise, we use ESP8266 to associate STM32F103C8 to the web. Here we will interface ESP8266 Wi-Fi module with our Blue Pill STM32F103C8 board along with send the information to a website page facilitated on ESP8266 webserver.

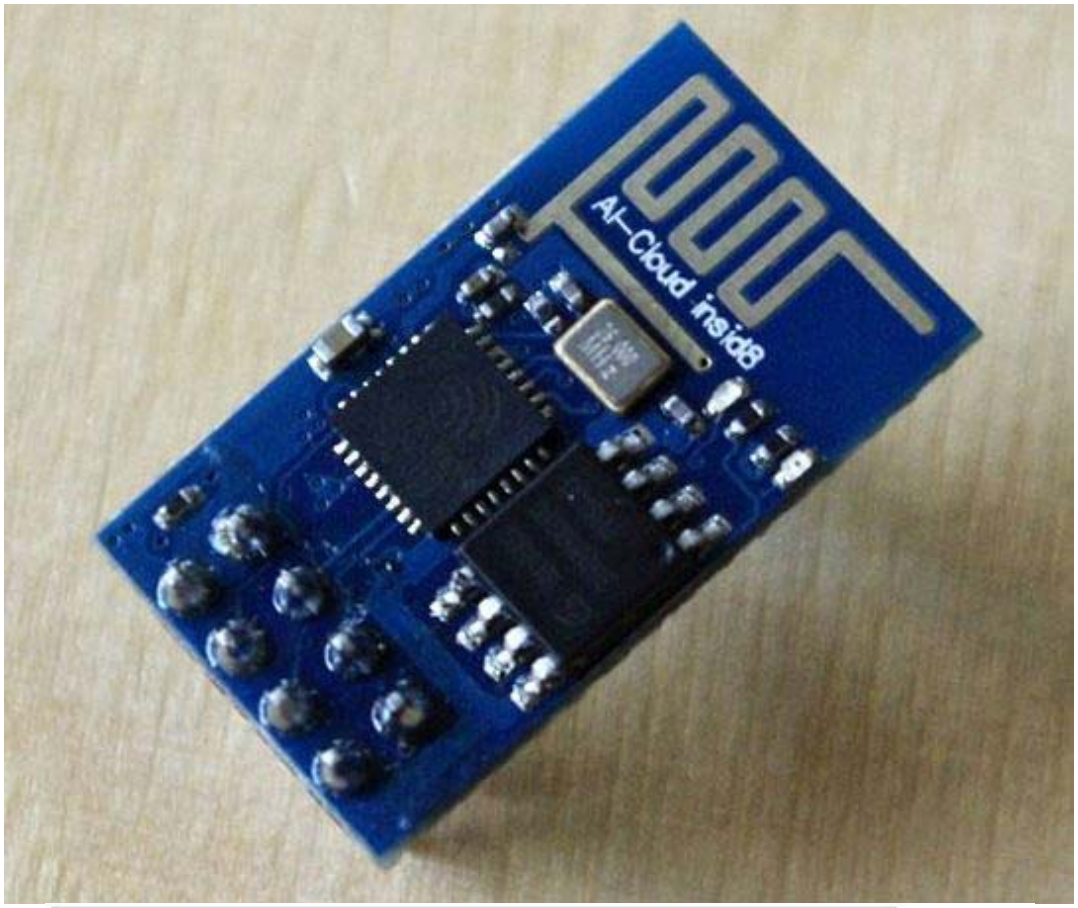
Parts Required

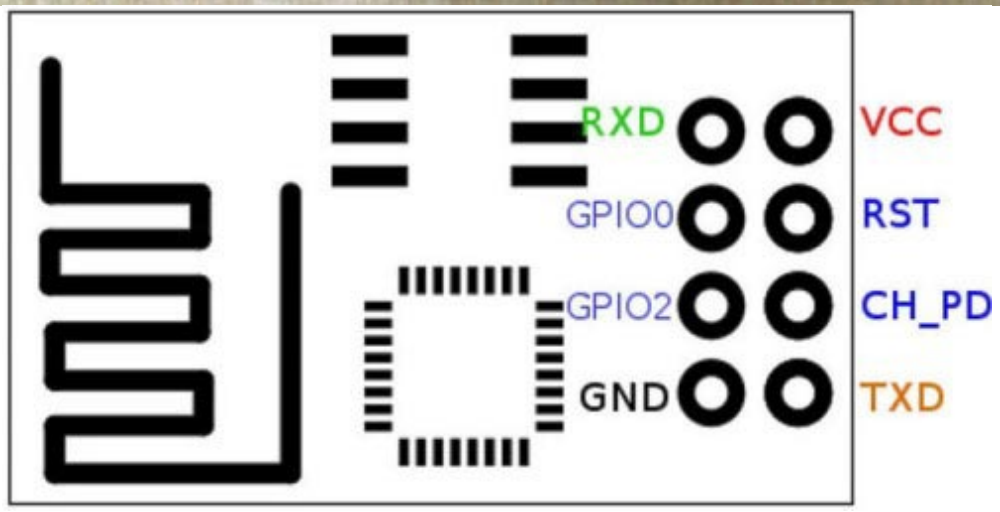
- Blue Pill STM32F103C8 board
- ESP8266 Wi-Fi module
- PC and Wi-Fi hotspot

ESP8266 Wi-Fi Module

A great many people call ESP8266 as a WIFI module, however it is really a microcontroller. ESP8266 is the name of the microcontroller created by Espressif Systems which is an organization based out of shanghai. This

microcontroller can perform WIFI related exercises thus it is broadly utilized as a WIFI module.





- GND, Ground (0 V)

- TX, Transmit information bit X
- GPIO 2, General-reason input/yield No. 2
- CH_PD, Chip shut down
- GPIO 0, General-reason input/yield No. 0
- RST, Reset
- RX, Receive information bit X
- VCC, Voltage (+3.3 V)

AT order is utilized for speaking with ESP8266. The table beneath shows some of helpful AT orders

AT COMMANDS	**USE**
AT	Acknowledgement returns 'OK'
AT+RST	RESTART module
AT+GMR	Shows FIRMWARE DETAILS
AT+CWMODE=1 or 2 or 3	Wi-Fi mode 1-Station, 2- AP

	,3-Both
AT+CWLAP	List the AP
AT+CWJAP=”SSID”,”PASSWORD”	Joins AP
AT+CWQAP	Quits AP
AT+CIFSR	Gets IP address
AT+CIPMUX= 0 or 1	Set multiple connection 0-Single ,1-multiple
AT+CIPSTART AT+CIPSTART=<type>,<address>,<port> AT+CIPSTART=<id>,<type>,<address>,<port>	Setup up TCP/UDP connection Address- Ip address Port-port address of ip Single Connection :Type-TCP, UDP Multiple Connection: Id-0 to 4,Type-TCP,UDP
AT+CIPSEND AT+CIPSEND=<length> AT+CIPSEND=<id>,<length>	Sends data Single Connection , Length - Length of data Multiple connection , Id =0 to 4
AT+CIPSTATUS	Gets the connection status
AT+CIPSERVER=<mode>,<port>	Set as Server 0-Server close, 1-Open port
AT+CIPCLOSE	Close TCP or UDP connection

We have made a progression of instructional exercises to comprehend the ESP8266, beginning from tenderfoot to propel level:

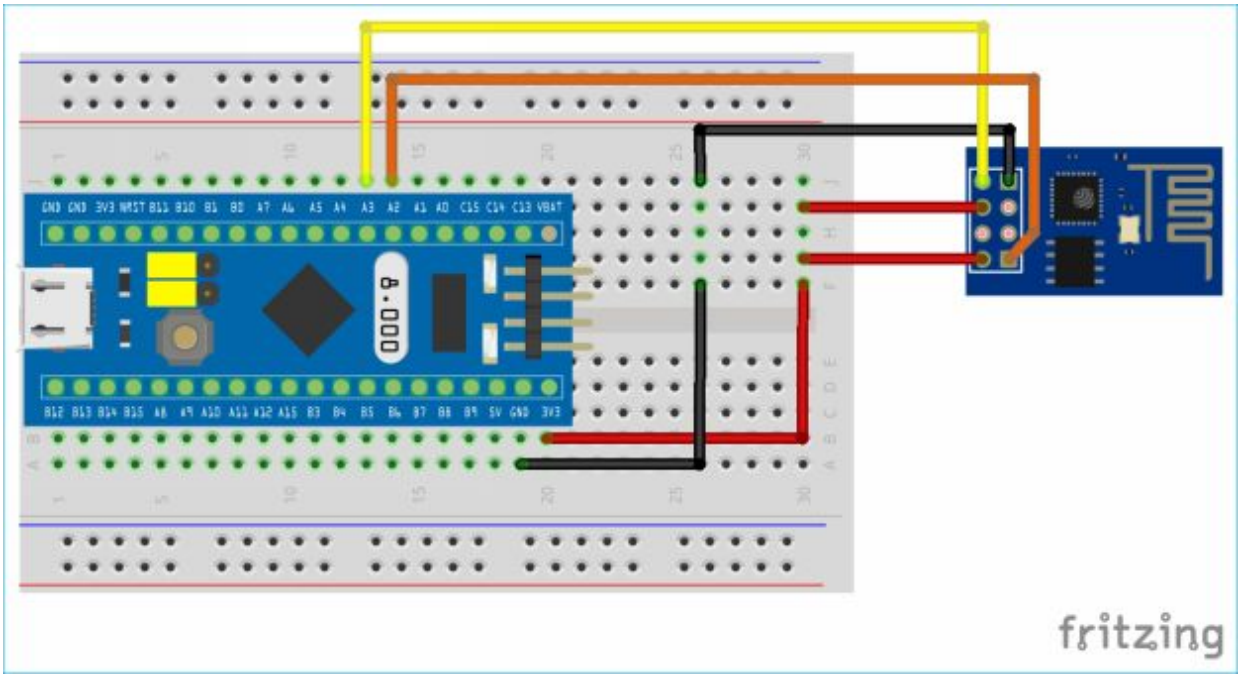
- Beginning with ESP8266 WiFi Transceiver (Part 1)

- Beginning with ESP8266 (Part 2): Using AT Commands

- Beginning with ESP8266 (Part 3): Programming ESP8266 with Arduino IDE and Flashing its Memory

Circuit Diagram and associations

Underneath schematic shows the association between STM32 and ESP8266 Wi-Fi module.

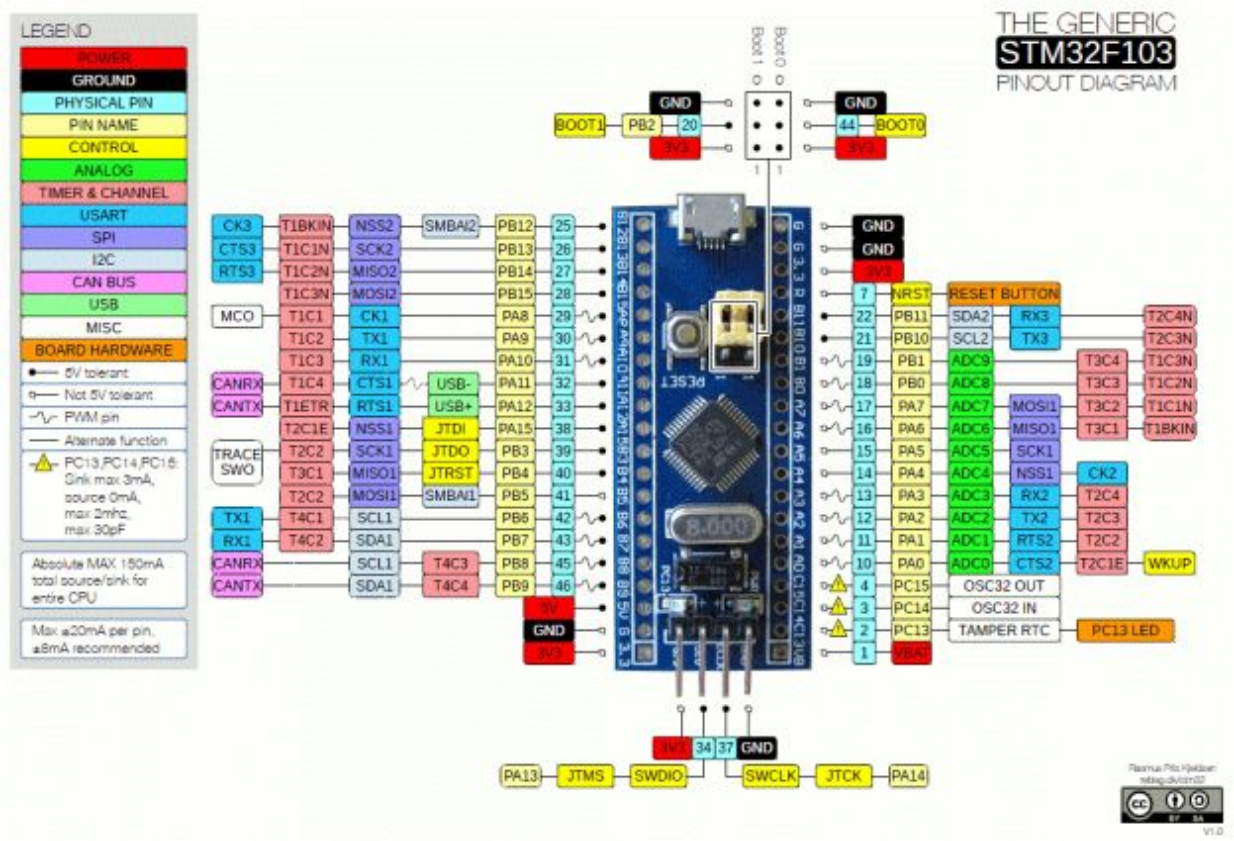


Allude underneath table to interface ESP8266 pins with STM32 pins:

ESP8266	STM32
VCC	3.3V
GND	G
CH_PD	3.3V
TX	PA3
RX	PA2

SMT32F103C8 has three arrangements of UART sequential correspondence. In the beneath picture you can see the accompanying pins for the equivalent:

Serial Port	Pins	Tolerant
Serial1 (TX1,RX1)	PA9,PA10 PB6,PB7	5V
Serial2 (TX2,RX2)	PA2,PA3	3.3V
Serial3 (TX3,RX3)	PB10,PB11	5V



ESP8266 utilizes sequential correspondence for associating with microcontroller. So here TX along with RX of ESP8266 are associated with serial2 port (PA2 along with PA3) of STM32 board.

Working and Code Interpretation

The working of interfacing ESP8266 with STM32 is basic. You can locate the total code given toward the finish of this instructional exercise.

We use arduino IDE to compose and transfer the code to STM32. Get familiar with utilizing Arduino IDE to program STM32 board.

First we have to do the circuit associations as appeared above in the circuit graph. In the wake of transferring the code, open Serial Monitor (Tools>>Serial Monitor) to see what's going on. You will view IP address on sequential screen, replicate the IP address from sequential screen and glue it in program and snap enter to see our website page. Ensure to interface your PC and ESP8266 module on a similar Wi-Fi organize.

Complete code is given toward the finish.

First we start sequential correspondence for sequential screen and for ESP8266, by utilizing following two explanations:

```
Serial.println(cmd);
Serial2.println(cmd);
```

NOTE: I have utilized pins (PA2, PA3) of Serial2 port of STM32 in light of the fact that it is 3.3V lenient.

At that point we have to prepare the ESP8266 by stopping any old associated AP by resetting it and setting Wi-Fi mode as both AP and STA

```
connect_wifi("AT",100);  //Sends AT command with time(Command for Acknowledgement)
connect_wifi("AT+CWMODE=3",100);   //Sends AT command with time (For setting mode of Wi-Fi)
connect_wifi("AT+CWQAP",100);  //Sends AT command with time (for Quit AP)
connect_wifi("AT+RST",5000);    //Sends AT command with time (For RESETTING WIFI)
```

At that point associate the ESP8266 with Wi-Fi arrange. You need to fill in your Wi-Fi subtleties as appeared in underneath code:

```
    connect_wifi("AT+CWJAP=\"Pramo\",\"pokemon08\"",7000);
//provide your WiFi username and password here
```

At that point we get the IP address of the ESP8266 module and show it on sequential screen by utilizing underneath code

```
    Serial2.println("AT+CIFSR");          //GET IP AT COMMAND
    if(Serial2.find("STAIP,"))            //This finds the STAIP that is the
STATIC IP ADDRESS of ESP8266
    Serial.print(IP);                          //prints IP address in Serial
monitor
```

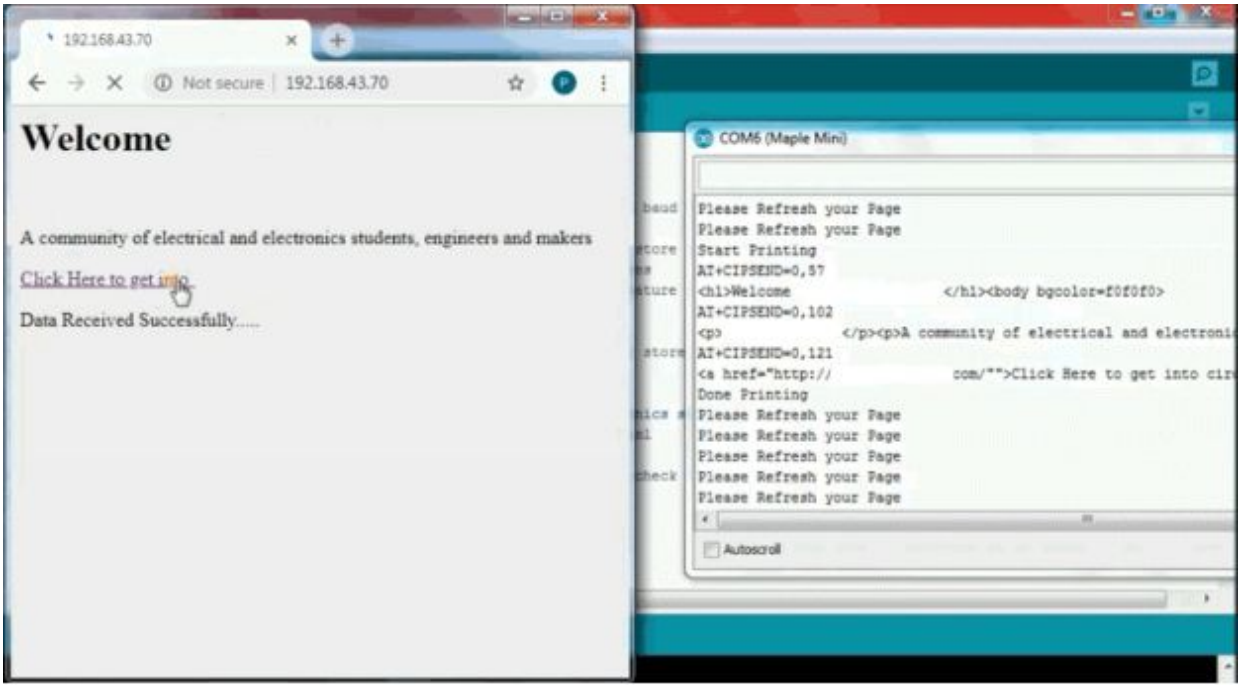
Next we will compose HTML code for the site page. To change over HTML code into Arduino code, you can utilize this connection.

```
    webpage = "<h1>Welcome</h1><body bgcolor=f0f0f0>"; //This is
the heading line with black font colour
    String name="<p>Hello World</p><p>A community of electrical
and electronics studen
    ts, engineers and makers</p>";
    String data="<p>Data Received Successfully.....</p>";    //These two
lines are of two paragraph
    webpage = "<a href=\"http://google.com/\"";
    webpage+="\">Click Here to get into google.com</a>"; //At last we
insert the hyperlink to link the website address
```

Next in void send() work, we have printed HTML utilizing sendwebdata work and shut the server association utilizing AT+CIPCLOSE=0

```
    void Send()                                   //This function contains data to
be sent to local server
    {
      webpage = "<h1>Welcome </h1><body bgcolor=f0f0f0>";
      sendwebdata(webpage);
      webpage=name;
      sendwebdata(webpage);
      delay(1000);
      webpage = "<a href=\"http://google.com/\"";
      webpage+="\">Click Here to get into goole.com</a>";
      webpage+=data;
      sendwebdata(webpage);
      Serial2.println("AT+CIPCLOSE=0");              //Closes the server
connection
    }
```

After the whole of the work, you can test the functioning by opening the ESP8266's IP in any internet browser and snap on the connection appeared on the page, Click Here to get into google.com, as beneath



In the wake of tapping the connection, you see a book on the site page saying "Information Received Successfully....."

Complete code is given underneath.

Code

```
//Interfacing ESP8266 Wi-Fi with STM32F103C8


    //NOTE: Serial is serial monitor with baud rate(9600)
//NOTE: Serial2 (TX2, RX2)is connected with ESP8266(RX,TX)respectively with baud rate (9600)
String webpage="";                          //String variable to store characters
int i=0,k=0,x=0;                            //integer variables
String readString;                          //using readString feature to read characters

boolean No_IP=false;                        //boolean variables
String IP="";                               //String variable to store data
char temp1='0';                             //character variable

String name="<p>Hello World</p><p>A community of electrical and electronics students, engineers and makers</p>";     //String with html notations
String data="<p>Data Received Successfully.....</p>";    //String with html



    void check4IP(int t1)                   //A function to check ip of ESP8266
{
 int t2=millis();
 while(t2+t1>millis())
 {
  while(Serial2.available()>0)
  {
   if(Serial2.find("WIFI GOT IP"))
   {
```

```
      No_IP=true;
     }
    }
  }
}
void get_ip()                                   //After cheacking ip ,this is a
function to get IP address
{
 IP="";
 char ch=0;
 while(1)
 {
  Serial2.println("AT+CIFSR");                 //GET IP AT COMMAND
  while(Serial2.available()>0)
  {
    if(Serial2.find("STAIP,"))                 //This finds the STAIP that is the
STATIC IP ADDRESS of ESP8266
    {
     delay(1000);
     Serial.print("IP Address:");
     while(Serial2.available()>0)
     {
      ch=Serial2.read();                   //Serial2 reads from ESP8266
      if(ch=='+')
      break;
      IP+=ch;
     }
    }
    if(ch=='+')
    break;
  }
  if(ch=='+')
  break;
  delay(1000);
 }
 Serial.print(IP);                          //prints IP address in Serial monitor
 Serial.print("Port:");
```

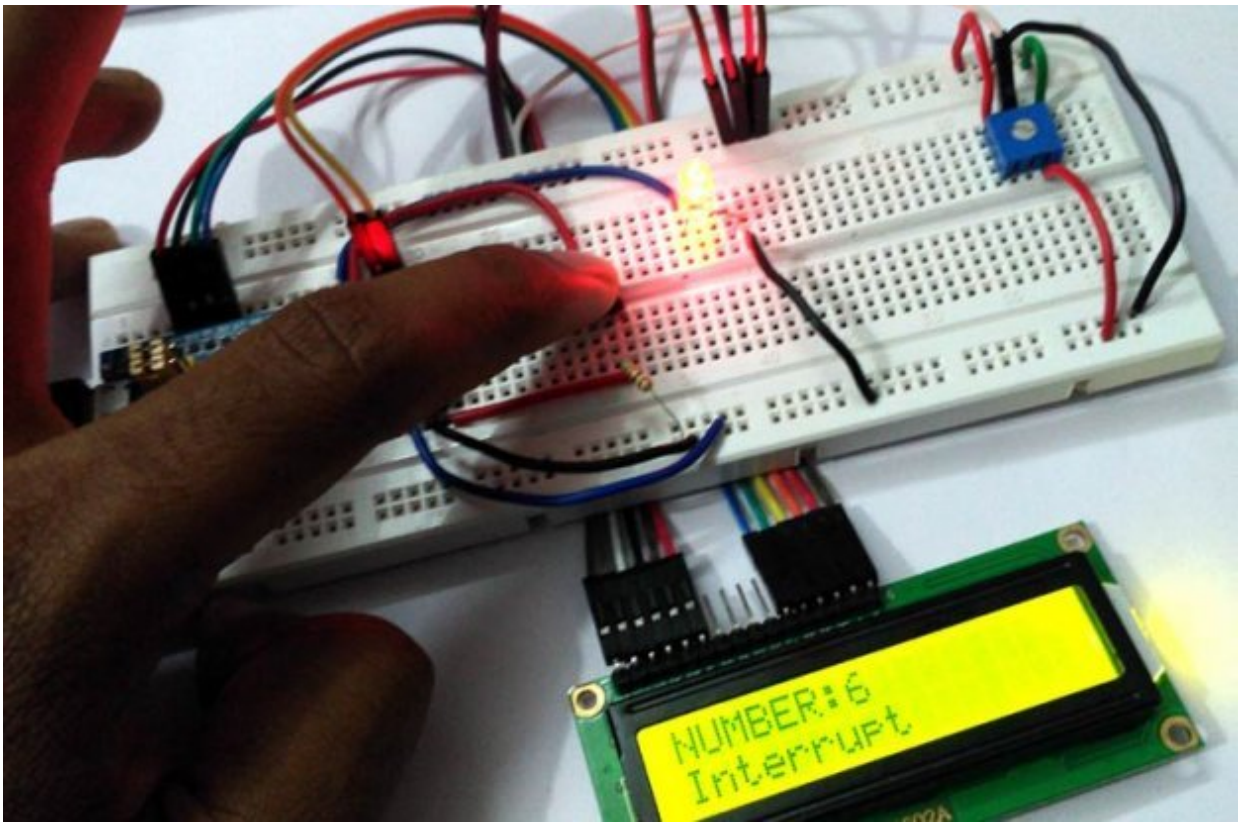
```
  Serial.println(80);
}

void connect_wifi(String cmd, int t)                    //This function is for connecting ESP8266 with wifi network by using AT commands
{
  int temp=0,i=0;
  while(1)
  {
    Serial.println(cmd);              //Sends to serial monitor
       Serial2.println(cmd);                          //sends to ESP8266 via serial communication
    while(Serial2.available())
    {
     if(Serial2.find("OK"))
     i=8;
    }
    delay(t);
    if(i>5)
    break;
    i++;
  }
  if(i==8)
  Serial.println("OK");
  else
  Serial.println("Error");
}

void wifi_init()                          //This function contains AT commands that passes to connect_wifi()
{
        connect_wifi("AT",100);                        //Sends AT command with time(Command for Achknowledgement)
     connect_wifi("AT+CWMODE=3",100);         //Sends AT command with time (For setting mode of Wifi)
       connect_wifi("AT+CWQAP",100);             //Sends AT command with time (for Quit AP)
     connect_wifi("AT+RST",5000);             //Sends AT command with time
```

```
(For RESETTING WIFI)
     check4IP(5000);
     if(!No_IP)
     {



          Serial.println("Connecting Wifi....");
          connect_wifi("AT+CWJAP=\"Pramo\",\"pokemon08\"",7000);
 //provide your WiFi username and password here



       }
     else
       {
       }
     Serial.println("Wifi Connected");
     get_ip();


       connect_wifi("AT+CIPMUX=1",100);                    //Sends AT
command with time (For creating multiple connections)
       connect_wifi("AT+CIPSERVER=1,80",100);              //Sends AT
command with time (For setting up server with port 80)
}
void sendwebdata(String webPage)                  //This function is used to
send webpage datas to the localserver
{
   int ii=0;
    while(1)
    {
     unsigned int l=webPage.length();
     Serial.print("AT+CIPSEND=0,");
     Serial2.print("AT+CIPSEND=0,");
     Serial.println(l+2);
     Serial2.println(l+2);
     delay(100);
      Serial.println(webPage);                    //sends webpage data to serial
```

```
monitor
      Serial2.println(webPage);                    //sends webpage data to serial2
ESP8266
     while(Serial2.available())
     {



         if(Serial2.find("OK"))
       {
        ii=11;
        break;
       }
     }
     if(ii==11)
     break;
     delay(100);
    }
}
void setup()
{
  Serial.begin(9600);              //begins serial monitor with baud rate 9600
  Serial2.begin(9600);               //begins serial communication with esp8266
with baud rate 9600 (Change according to your esp8266 module)
  wifi_init();
  Serial.println("System Ready..");
}

void loop()
{
 k=0;
 Serial.println("Please Refresh your Page");
 while(k<1000)
 {
  k++;
 while(Serial2.available())
 {
  if(Serial2.find("0,CONNECT"))
```

```
   {
    Serial.println("Start Printing");
    Send();
    Serial.println("Done Printing");
    delay(1000);
   }
  }
  delay(1);
 }
}

void Send()                                   //This function contains data to be sent
to local server
{
    webpage = "<h1>Welcome</h1><body bgcolor=f0f0f0>";
    sendwebdata(webpage);
    webpage=name;
    sendwebdata(webpage);
    delay(1000);
    webpage = "<a href=\"http://google.com/\"";
    webpage+="\">Click Here to get into google.com</a>";
    webpage+=data;
    sendwebdata(webpage);
      Serial2.println("AT+CIPCLOSE=0");                  //Closes the server
connection
}
```

4. Step by step instructions to Use Interrupts in STM32F103C8



Hinders is a system by which an I/O or a guidance can suspend the typical execution of processor and gets itself adjusted like it has most elevated need. Like for instance, a processor doing a typical execution can likewise constantly screen for an occasion or a hinder to occur. That is the point at which an outside hinder is occurred (like from some sensor) at that point the processor stop its typical execution and first serves the hinder and afterward proceed with its ordinary execution.

Here in this task, for understanding the Interrupts in STM32F103C8, we will utilize press button as outside intrude. Here we will augment a number from 0 and show it on 16x2 LCD, and at whatever point the press button is squeezed the drove turns ON and the LCD show shows INTERRUPT. Driven turns off when the catch is discharged.

Kinds of Interrupts and ISR

Hinders can be comprehensively characterized into two sorts:

Equipment Interrupts: If the sign to the processor is from some outer gadget, for example, catch or sensor or from some other equipment gadget which creates a sign and advise processor to do specific undertaking present in ISR is known as equipment interferes.

Programming Interrupts: The interferes with which are created by the product directions.

Interfere with Service Routine

Interfere with Service Routine or an Interrupt handler is an occasion that has little arrangement of directions in it and when a hinder is happened the processor initially executes these code that is available in ISR and afterward proceed with the assignment which it was doing before the intrude.

Sentence structure for Interrupt in STM32

ISR has following sentence structure attachInterrupt (digitalPinToInterrupt(pin), ISR, mode) in Arduino and the equivalent can likewise be utilized in STM32 as we use arduino IDE to transfer code.

- **digitalPinToInterrupt(pin):** Like in Arduino board Uno we have pins 2,3 and in super we have 2,3,18,19,20,21 for intrudes. In STM32F103C8 we likewise have intrude on pins any GPIO pins can be utilized for interferes. We just to indicate the information pin that we are utilizing for interfere. In case, while utilizing more than 1 hinders simultaneously we may need to follow a few limitations.

- **ISR:** It an Interrupt handler work that is considered when an outside hinder is happened. It has no contentions and void bring type back.

- **Mode:** Type of change to trigger the hinder

1. **Ascending:** To trigger a hinder when the pin travels from LOW to HIGH.

2. **FALLING:** To trigger a hinder when the pin travels from HIGH to LOW.

3. **CHANGE:** To trigger a hinder when the pin travels either from LOW to HIGH or HIGH to LOW (i.e., when the pin changes).

A few conditions while utilizing hinder

- Intrude on Service Routine capacity (ISR) must be as short as could reasonably be expected.

- Deferral () work doesn't work inside ISR and must be evaded.

Segments Required

- STM32F103C8

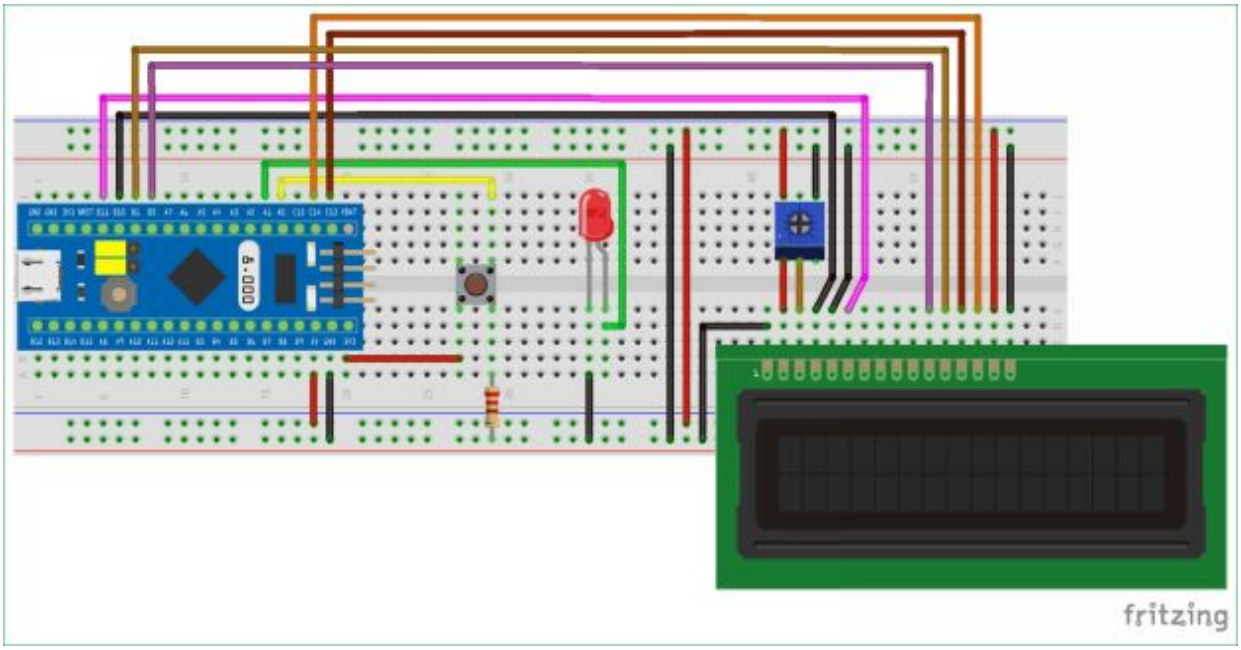
- Press button

- Driven

- Resistor (10K)

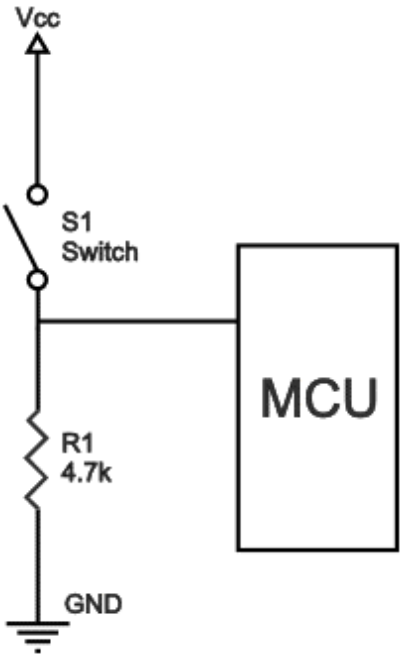
- LCD (16x2)

Circuit Diagram and Connections



One side of the press button pin is associated with 3.3V of STM32 and the opposite side is associated with input pin (PA0) of STM32 by means of a draw down resistor.

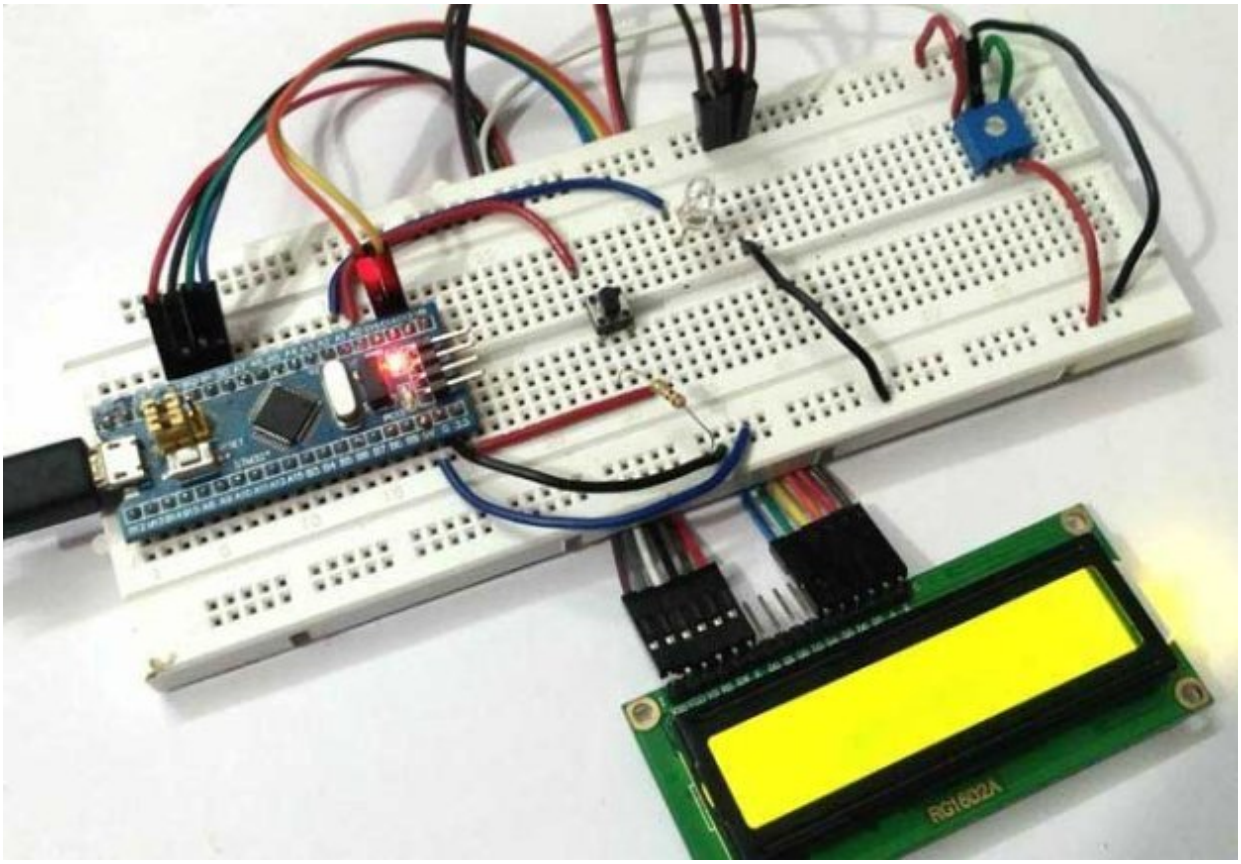
Pull Down resistor is utilized so that microcontroller will possibly get either HIGH otherwise LOW at its info when catch is compressed or discharged. Something else, without pull down resistor, MCU may get befuddle and feed some arbitrary skimming esteems to the info.



Association among STM32F103C8 and LCD

The accompanying table shows the pin association between LCD (16X2) and the STM32F103C8 microcontroller.

STM32F103C8	**LCD**
GND	VSS
+5V	VDD
To Potentiometer Centre PIN	V0
PB0	RS
GND	RW
PB1	E
PB10	D4
PB11	D5
PC13	D6
PC14	D7
+5V	A
GND	K



Programming STM32F103C8 for hinders

Program for this instructional exercise is straightforward and given toward the finish of this instructional exercise. We needn't bother with FTDI developer to program STM32, basically associate your PC to USB port of STM32 and begin programming with Arduino IDE. Well known with programming STM32 across Universal Serial Bus port.

As we said that here in this instructional exercise we are going to increase a number from 0 and show it in on 16x2 LCD and at whatever point a press button is squeezed the drove goes ON and LCD show shows 'Interfere'.

First characterize LCD pins associations with STM32. You can alter it according to your prerequisites.

```
const int rs= PB10,en= PB11,d4= PB0,d5= PB1,d6= PC13,d7= PC14;
```

Next, we incorporate the header record for the LCD show. This calls the library which contains the code for how the STM32 ought to speak with the LCD. Likewise ensure the capacity LiquidCrystal is called with the pin names that we simply characterized previously.

```
include<LiquidCrystal.h>
LiquidCrystal lcd (rs,en,d4,d5,d6,d7);
```

Worldwide factors are utilized to pass information among ISR and the principle program. We proclaim the variable ledOn as unpredictable and furthermore as Boolean to determine True or False.

```
volatile boolean ledOn = false;
```

Inside the void arrangement() work, we will show an introduction message and clear it following 2 seconds.

```
lcd.begin(16,2);
lcd.print("Hello Wolrd");
delay(2000);
lcd.clear();
```

Again in same void arrangement () work, we have to indicate the info and yield pins. We set pin PA1 for yield to LED and PA0 for contribution from press button.

```
pinMode(PA1,OUTPUT)
pinMode(PA0,INPUT)
```

We are likewise going to increase a number, so pronounce a variable with esteem zero.

```
    int i = 0;
```

Presently the significant piece of the code is attachInterrupt() work, it is additionally included inside the void arrangement()

```
    attachInterrupt(digitalPinToInterrupt(PA0),buttonPressed,CHANG
E)
```

We indicated the pin PA0 for outside intrude, and buttonPressed is the capacity which is to be called when there is CHANGE (LOW to HIGH or HIGH to LOW) in PA0 pin. You can likewise use few other capacity name, pin and mode as per necessity.

Inside the void circle() we increase a number (I) from zero and print the number in LCD(16x2).

```
    lcd.clear();
    lcd.print("NUMBER:");
    lcd.print(i);
    ++i;
    delay(1000);
```

The superior part is making an interfere with handler work as per the name that we utilized in the attachInterrupt() work. We utilized buttonPressed so here we have made a capacity void buttonPressed()

```
    void buttonPressed()
    {
 if(ledOn)
 {
     ledOn=false;
     digitalWrite(PA1,LOW);
 }
 else
 {
     ledOn = true;
```

```
    digitalWrite(PA1,HIGH);
    lcd.setCursor(0,1);
    lcd.print("Interrupt");
 }
   }
```

Working of this buttonPressed() ISR:

As indicated by the estimation of ledOn variable, the Light Emitting Diode turns ON along with OFF.

BUTTON STATE	ledOn(Value)	LED(Red)	LCD(16x2)
UNPRESSED	False	OFF	-
PRESSED	True	ON	Shows ‘’INTERRUPT’

On the off chance that the ledOn esteem is bogus, at that point LED stays killed and in the event that the ledOn esteem is True, at that point LED turns on and LCD show shows 'Intrude' on it.

NOTE: There might be switch debounce impact now and then and it might check numerous trigger when pushbutton is squeezed, this is on the grounds that few spikes in voltage because of mechanical explanation of exchanging press button. This can be reduced by presenting RC channel.

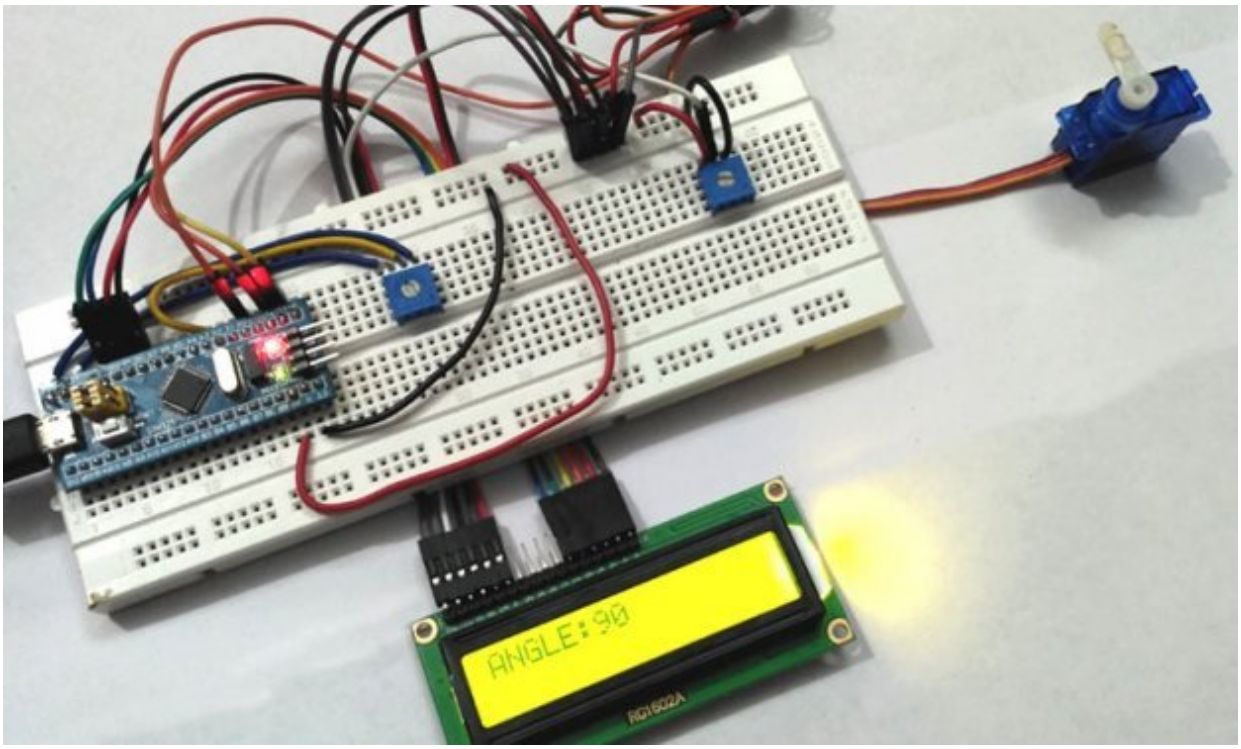
Code

```
//INTERRUPTS IN STM32F103C8


const int rs= PB10,en= PB11,d4= PB0,d5= PB1,d6= PC13,d7= PC14;     // declaring pin names and pin numbers of lcd
#include<LiquidCrystal.h>                                  //  including lcd display library
```

```
LiquidCrystal lcd (rs,en,d4,d5,d6,d7);                    //  setting lcd and its
parameters
volatile boolean ledOn = false;                          //  variable declared as
global

void setup()

{
  lcd.begin(16,2);                                       //  setting LCD as 16x2 type
  lcd.print("Hello world");                               //  puts Hello world IN LCD
  delay(2000);                                           //  delay time
  lcd.clear();                                           //  clears lcd display
   pinMode(PA1,OUTPUT);                                          //  set pin PA1 as
output
  pinMode(PA0,INPUT);                                    //  set pin PA0 as input
   int i = 0;                                            //  declare variable i and
initiliaze with 0

  attachInterrupt(PA0,buttonPressed,CHANGE);                 //  function for
creating external interrupts



     }
void loop()                                              //  void loops runs continuously
{
  lcd.clear();                                           //  clears lcd display
  lcd.print("NUMBER:");                                          //  puts NUMBER: in
LCD display
  lcd.print(i);                                          //  prints the values of i in LCD
  ++i;                                                   //  increments value of i
  delay(1000);                                           //  delays time

}

void buttonPressed()                                     //
{
  if(ledOn)                                              //  if statement depends on
```

```
LedOn value
  {
    ledOn=false;                                  //  Makes ledOn false if it is
True
    digitalWrite(PA1,LOW);                              //  digital writs the low
vale to PA1 pin makes led OFF
  }
  else
  {
    ledOn = true;                                 //  Makes ledOn True if it is
False
     digitalWrite(PA1,HIGH);                                //  digital writs the
HIGH vale to PA1 pin makes led ON
    lcd.setCursor(0,1);                           //  sets cursor at first column
and second row
     lcd.print("Interrupt");                             //  puts INTERRUPT in
LCD display
  }
}
```

5. Interfacing Servo Motor with STM32F103C8



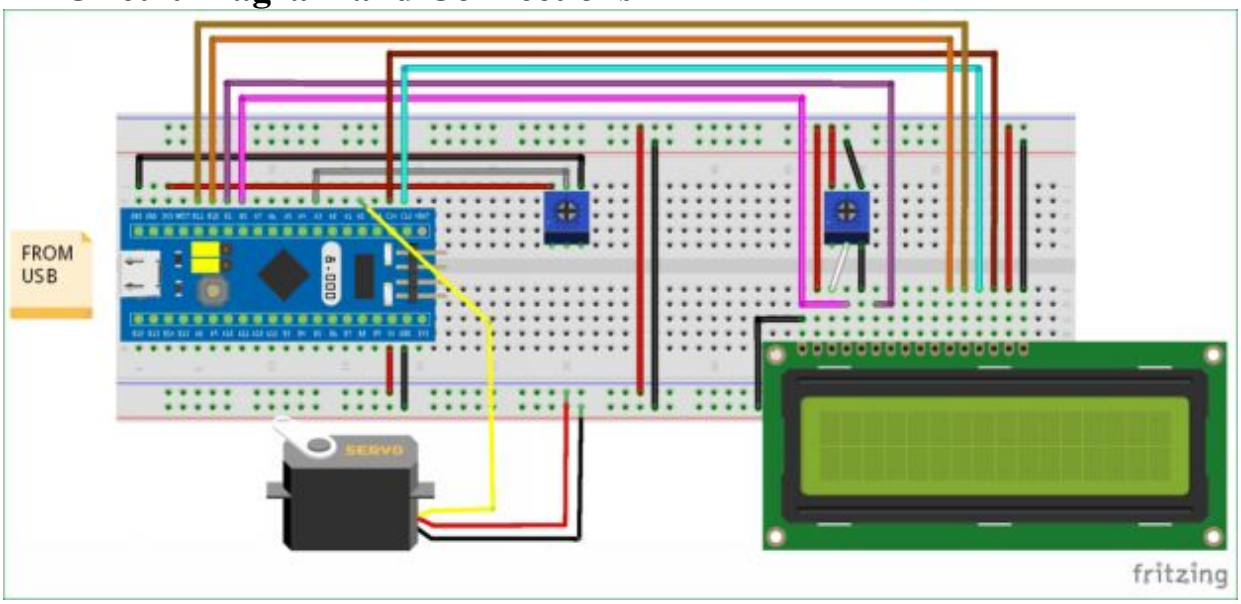
In hardware, Servo engines are for the most part utilized in Robotics Projects as a result of their exactness and simple dealing with. Servo engines are littler in size and they are extremely compelling and vitality proficient. They give high torque and can be utilized to lift or push loads as per engines particular. In this instructional exercise we will find out about Servo Motor and How to interface Servo with STM32F103C8 board. A potentiometer is additionally interfaced to change the situation of the servo engine's pole, and a LCD to show the point esteem.

Segments Required

- STM32F103C8 (Blue Pill) Board
- Servo Motor (SG90)
- LCD(16x2)

- Potentiometer
- Breadboard
- Jumper Wires

Circuit Diagram and Connections



SMT32F103C8 Pin Details

In STM32F103C8, we have 10 ADC pins (PA0-PB1), and here we utilize just one pin (PA3) for analogread() for setting shaft position of engine by potentiometer. Likewise among 15 PWM pins of STM32 (PA0, PA1, PA2, PA3, PA6, PA7, PA8, PA9, PA10, PB0, PB1, PB6, PB7, PB8, PB9), one pin will be utilized for giving heartbeats to the Servo engine's PWM pin(usually it is orange in shading).

You can get familiar with PWM and ADC by perusing beneath to definite articles:

- The most effective method to utilize ADC in STM32F103C8

- Heartbeat width Modulation (PWM) in STM32F103C8

Association among STM32F103C8 and LCD

STM32F103C8	LCD
GND	VSS
+5V	VDD
To Potentiometer Centre PIN	V0
PB0	RS
GND	RW
PB1	E
PB10	D4
PB11	D5
PC13	D6
PC14	D7
+5V	A
GND	K

Association between Servo engine and STM32F103C8

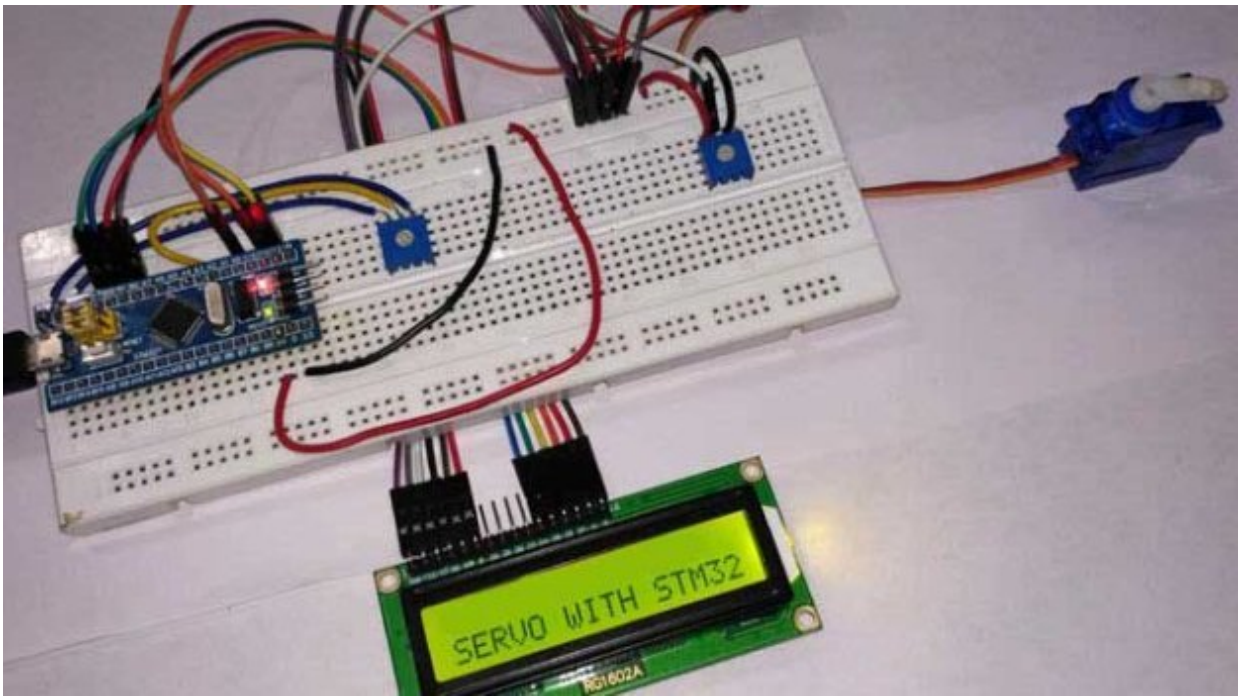
STM32F103C8	**SERVO**
+5V	RED (+5V)
PA0	ORANGE (PWM pin)
GND	BROWN (GND)

Potentiometers Connections

We have utilized TWO potentiometers here

1. The potentiometer on the privilege is utilized to differ the LCD differentiate. It has three pins, left pin is for +5V and right is for GND and focus pin is associated with V0 of the LCD.

2. The potentiometer on the left is utilized to differ the pole position of servo engine by controlling the simple information voltage, the left pin has input 3.3V and right has GND and focus yield is associated with (PA3) of STM32



Programming STM32 for Servo Motor

Like our past instructional exercise, we modified the STM32F103C8 with Arduino IDE through USB port without utilizing FTDI developer. We can continue programming it like an Arduino. Complete code is given underneath toward the finish of task.

First we have included library documents for servo and LCD capacities:

```
#include<Servo.h>
#include<LiquidCrystal.h>
```

At that point pronounced pins for LCD show and introduced it. Likewise pronounced hardly any different factors for PWM and potentiometer:

```
const int rs = PB0, en = PB1, d4 = PB10 , d5 = PB11 , d6 = PC13, d7 = PC14;
LiquidCrystal lcd(rs,en,d4,d5,d6,d7);
int servoPin = PA0;
int potPin = PA3;
```

Here we have made variable servo with datatype Servo and connected it to recently pronounced PWM pin.

```
Servo servo;
servo.attach(servoPin);
```

At that point read Analog incentive from pin PA3 as it is an ADC pin it changes over simple voltage (0-3.3) into computerized structure (0-4095)

```
analogRead(potPin);
```

As the advanced yield is 12-piece goals, we have to get values in scope of degree (0-170), it partitions ADC (0-4096) esteem as per max edge 170 deg so we isolate with 24.

```
angle = (reading/24);
```

Underneath articulation makes the servo engine to turn the pole at edge given.

```
servo.write(angle);
```

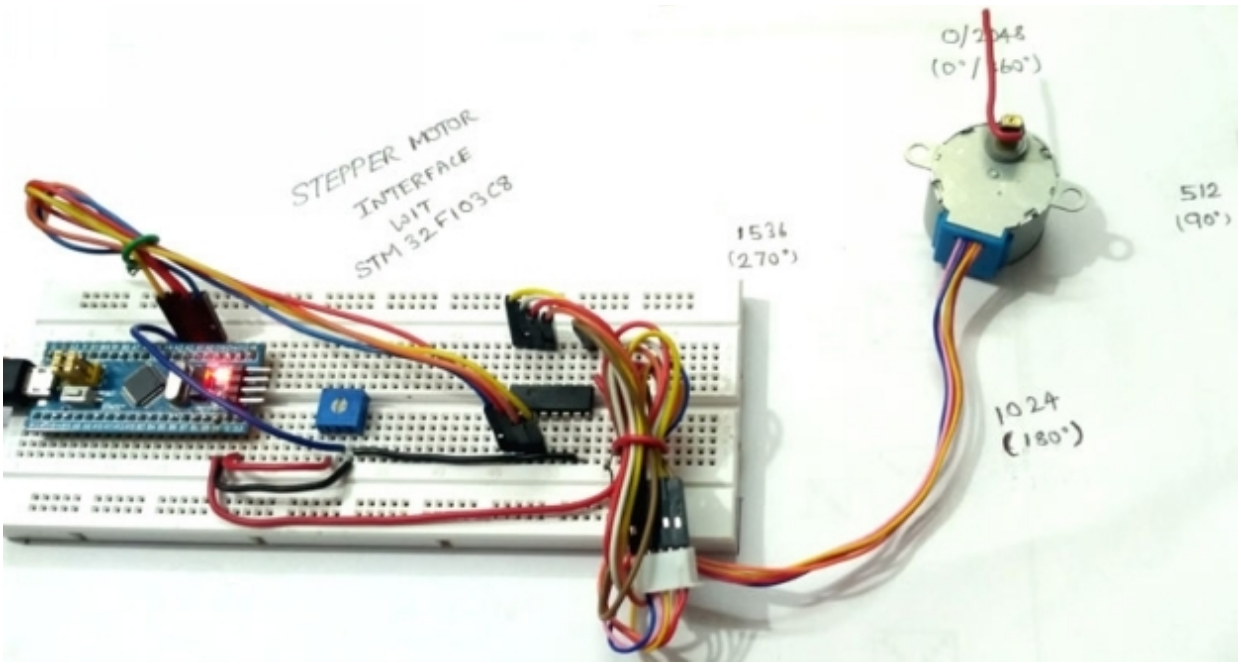
Complete code is given beneath and very much clarified by remarks.

Code

```
//INTERFACE SERVO WITH STM32


#include<Servo.h>                                                    //including servo
library
#include<LiquidCrystal.h>                                            //including LCD
display library
const int rs = PB0, en = PB1, d4 = PB10 , d5 = PB11 , d6 = PC13, d7 =
PC14;   //declaring pin names and pin numbers of lcd
LiquidCrystal lcd(rs,en,d4,d5,d6,d7);                               //setting lcd and its
paramaters

int servoPin = PA0;                          //declare and initialize pin for servo
output PWM
int potPin = PA3;                           //potentiometer ADC input

Servo servo;                                    // creating variable servo with
datatype Servo

void setup()

{
  lcd.begin(16,2);                                                  //setting lcd as 16x2
  lcd.setCursor(0,0);                                               //setting cursor at first
row and first column
  lcd.print("Hello world");                                         //puts Hello world in
LCD
  lcd.setCursor(0,1);                                               //setting cursor at
second row and first column
  lcd.print("SERVO WITH STM32");                                    //puts
SERVO WITH STM32 in LCD
  delay(3000);                                                      // delays for 3 seconds
  lcd.clear();                                                     //clears lcd display
    servo.attach(servoPin);     //it connects pin PA0 with motor as control
```

```
feedback by providing pulses



    }
void loop()
{
 lcd.clear();                                          //clears lcd
 int angle;                                            //declare varible angle as
int
 int reading;                                          //declare varible reading
as int
 reading = analogRead(potPin);                                  //read analog
value from pin PA3
 angle = (reading/24);                                 //it divides ADC the
value according to max angle 170 deg
 servo.write(angle);                                   //it puts angle value
at servo
 lcd.setCursor(0,0);                                     //setting cursor at
first row and first column
 lcd.print("ANGLE:");                                   //puts ANGLE in
LCD
 lcd.print(angle);                                     //puts value at angle
 delay(100);                                           //delay in time
}
```

6. Interfacing Stepper Motor with STM32F103C8



Stepper engine is brushless DC engine, which can be pivoted in little points, these edges are called steps. Mostly stepper engine utilize 200 stages to finish 360 degree pivot, implies its turn 1.8° per step. Stepper engine is utilized in numerous gadgets which needs exact rotational development like robots, recieving wires, hard drives and so on. We can turn stepper engine to a specific edge by giving it appropriate directions. For the most part two kinds of stepper engines are accessible, Unipolar and Bipolar. Unipolar is simpler to work, control and furthermore simpler to get. Here in this instructional exercise we are interfacing Stepper Motor with STM32F103C8 (Blue pill) board.

Elements Necessary

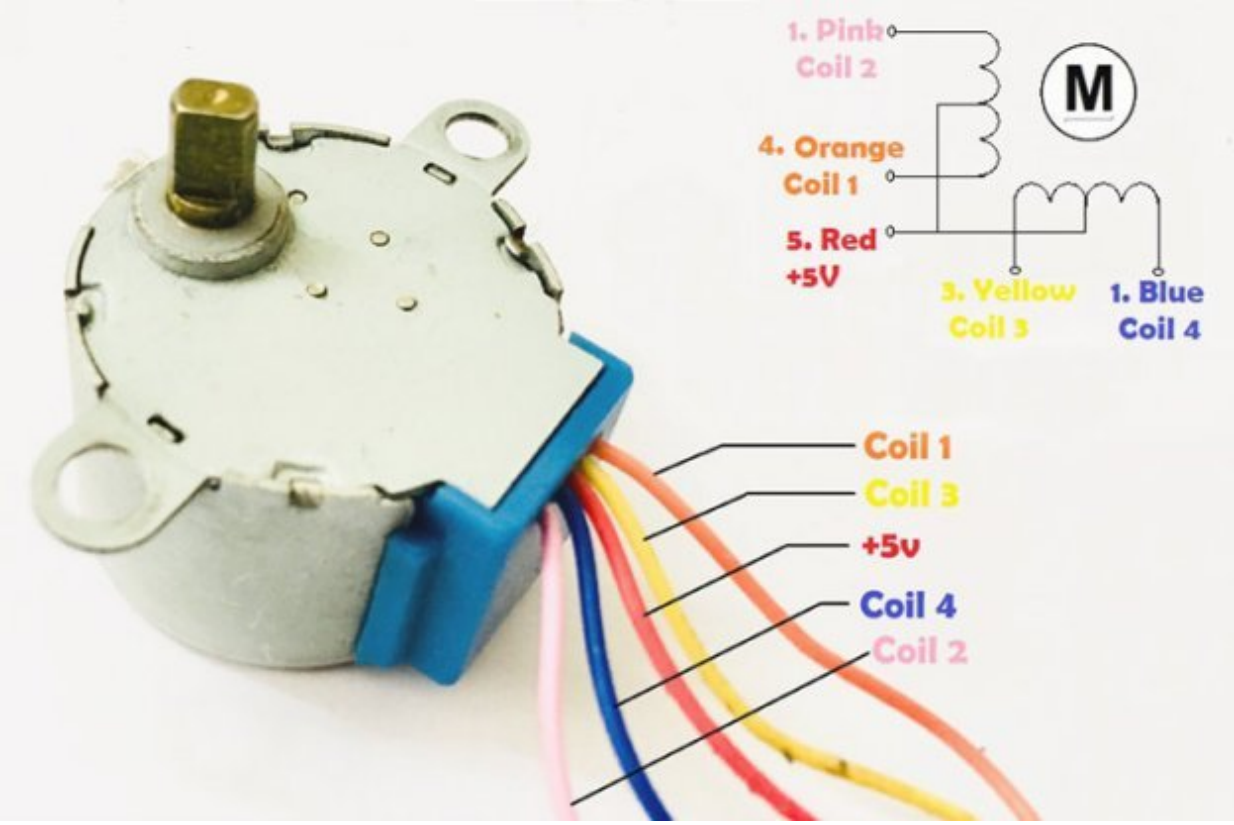
- Stepper Motor(28BYJ-48)

- STM32F103C8 (Blue pill)

- Potentiometer 10k
- ULN2003 IC
- Jumper wires
- Breadboard

Stepper Motor (28BYJ-48)

28BYJ-48 is a Unipolar Stepper engine which needs 5V supply. The engine has a 4 loop unipolar course of action and each curl is appraised for +5V subsequently it is moderately simple to control with any microcontrollers like Arduino ,Raspberry Pi likewise STM32.But we need a Motor Drive IC like ULN2003 to drive it, since stepper engines expend high present and it might harm microcontrollers.



Another significant information to see is the Stride Angle: 5.625°/64. This implies the engine when works in 8-advance arrangement will move 5.625 degree for each progression and it will make 64 strides (5.625*64=360) to finish one full revolution. Different determinations are given in datasheet beneath:

Rated voltage :	5VDC
Number of Phase	4
Speed Variation Ratio	1/64
Stride Angle	5.625°/64
Frequency	100Hz
DC resistance	50Ω±7%(25℃)
Idle In-traction Frequency	> 600Hz
Idle Out-traction Frequency	> 1000Hz
In-traction Torque	>34.3mN.m(120Hz)
Self-positioning Torque	>34.3mN.m
Friction torque	600-1200 gf.cm
Pull in torque	300 gf.cm
Insulated resistance	>10MΩ(500V)
Insulated electricity power	600VAC/1mA/1s
Insulation grade	A
Rise in Temperature	<40K(120Hz)
Noise	<35dB(120Hz,No load,10cm)
Model	28BYJ-48 – 5V

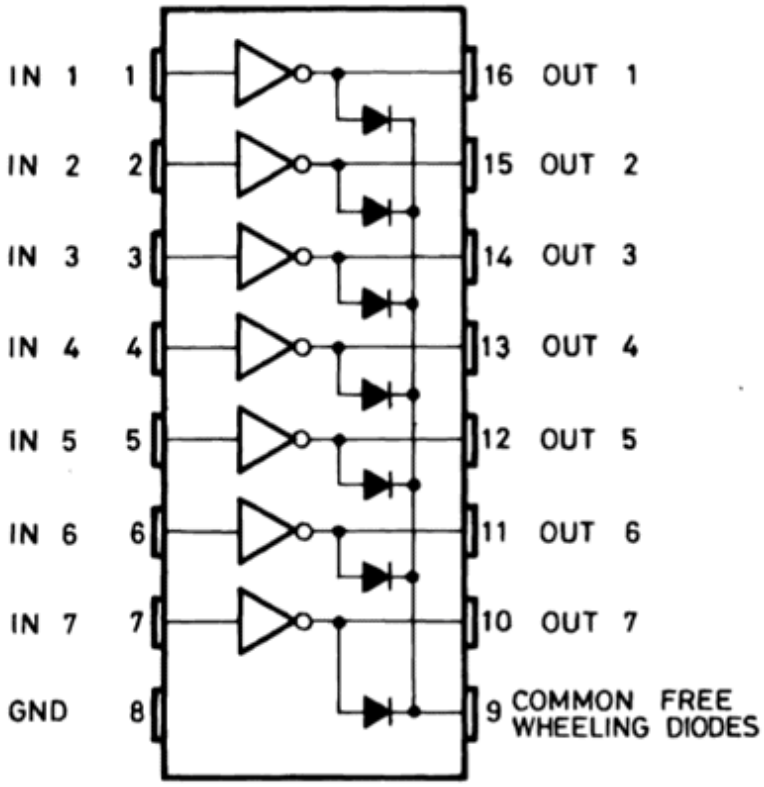
Additionally check interfacing with Stepper Motor with different Microcontrollers:

- Interfacing Stepper Motor with Arduino Uno
- Stepper Motor Control with Raspberry Pi
- Stepper Motor Interfacing with 8051 Microcontroller
- Interfacing Stepper Motor with PIC Microcontroller

Stepper engine can likewise be controlled with no Microcontroller, view this Stepper Motor Driver Circuit.

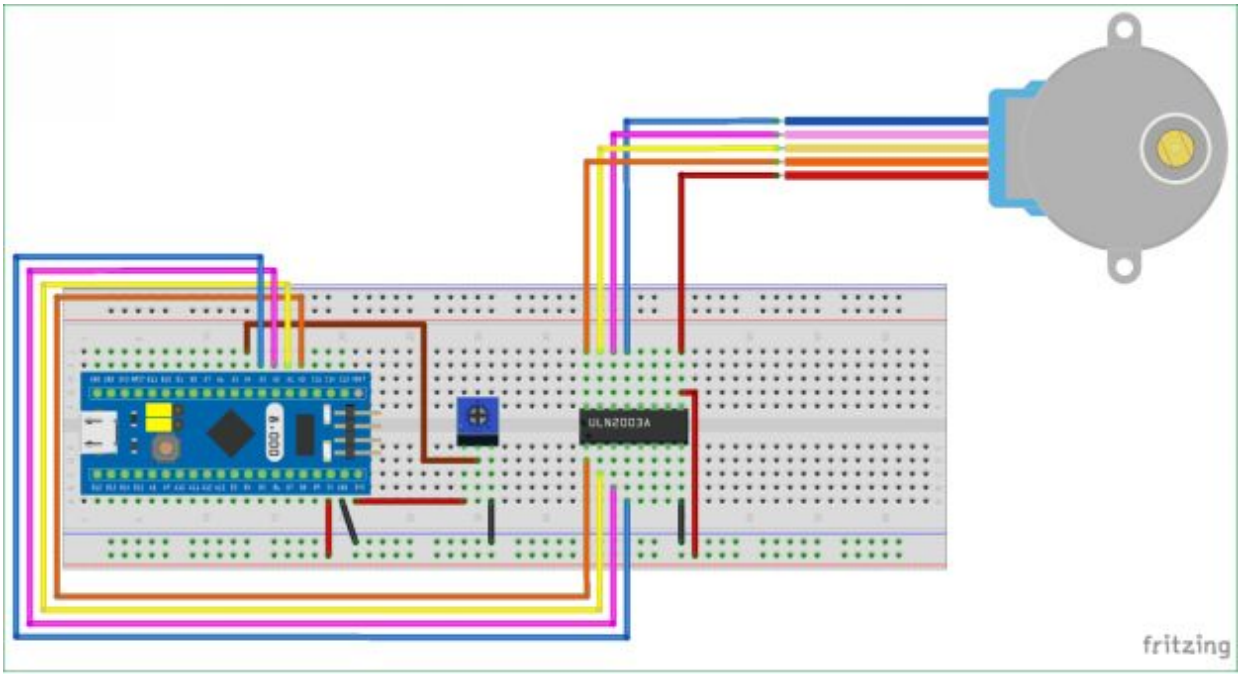
ULN2003 Motor Driver IC

It is utilized to drive the engine as per beats got from microcontroller. The following is the image outline of ULN2003:



Pins (IN1 to IN7) are input sticks and (OUT 1 to OUT 7) are comparing yield pins. COM is given Positive source voltage required for yield gadgets. Further associations for stepper engine are given underneath in circuit outline segment.

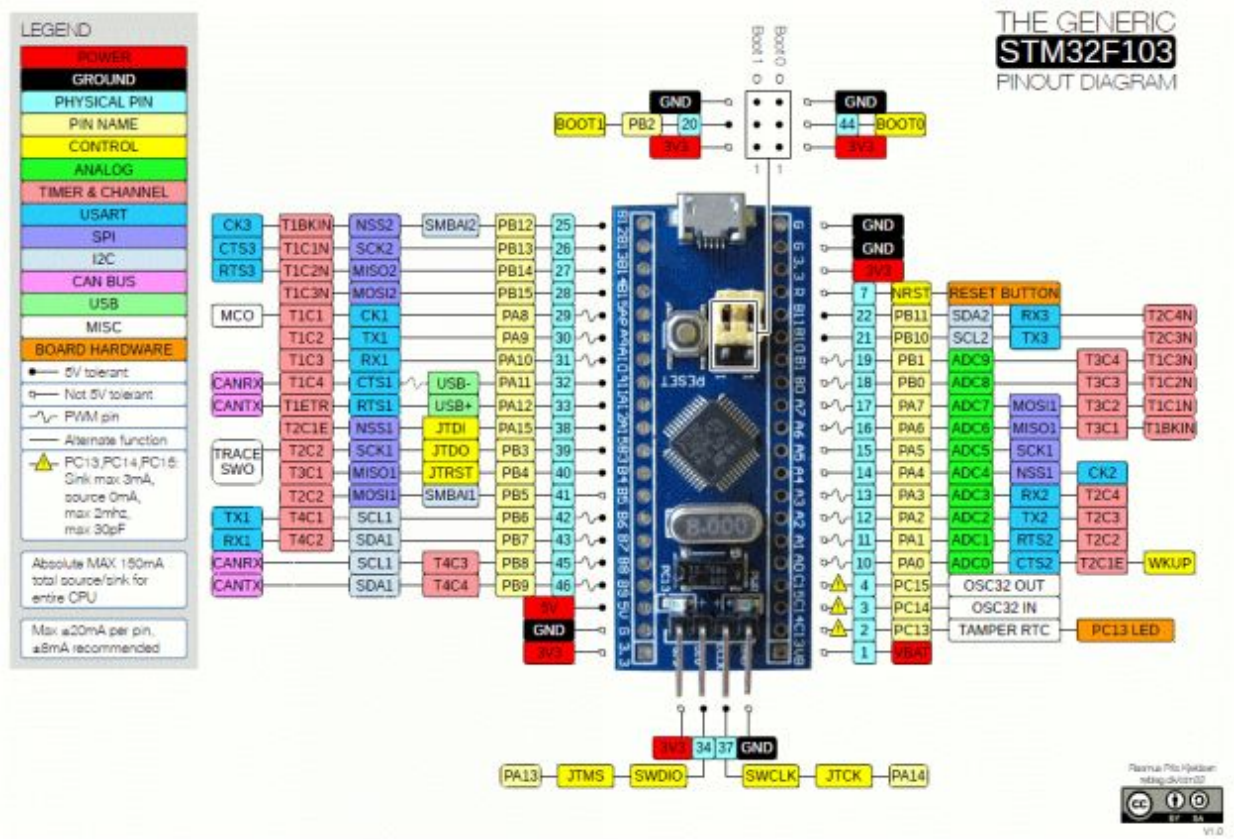
Circuit Diagram and Connections



The following is the associations clarification for above circuit graph.

STM32F103C8 (Blue Pill)

As must be clear in the beneath graph, the PWM pins are shown in wave position (~), there are 15 such pins which can be utilized for beat yield to stepper engine. We need just four pin, we use (PA0 toPA3).



STM32F103C8 with ULN2003 Motor Driver IC

Pins (PA0 to PA3) are considered as yield sticks that are associated with input pins (IN1-IN4) of the ULN2003 IC.

PINS OF STM32F103C8	PINS OF ULN2003 IC
PA0	IN1
PA1	IN2
PA2	IN3
PA3	IN4
5V	COM
GND	GND

ULN2003 IC with Stepper Motor (28BYJ-48)

The yield pins (OUT1-OUT4) of ULN2003 IC are associated with the stepper engines pins (Orange, Yellow, Pink, and Blue).

PINS OF ULN2003 IC	PINS OF STEPPER MOTOR
OUT1	ORANGE
OUT2	YELLOW
OUT3	PINK
OUT4	BLUE
COM	RED

STM32F103C8 with Potentiometer

A potentiometer is utilized as to set speed of the stepper engine.

POTENTIOMETER	STM32F103C8
LEFT (INPUT)	3.3
CENTRE(OUTPUT)	PA4
RIGHT(GND)	GND

Pivoting Stepper Motor with STM32F103C8

The following is not many strides to work the Stepper Motor:

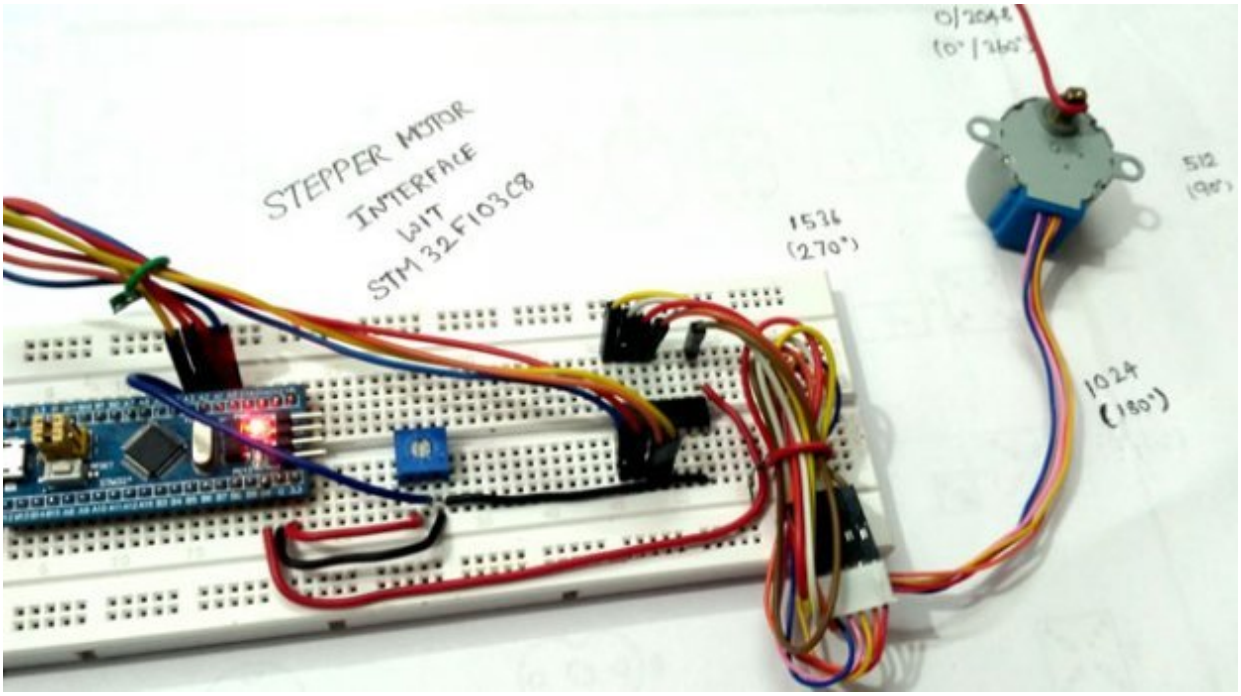
- Set the speed of stepper engine by fluctuating potentiometer.

- At that point physically enter ventures for pivot either in clockwise (+values) or anticlockwise course(- values) by means of SERIAL MONITER present in ARDUINO IDE (Tools->Serial screen) or CTRL+SHIFT+M.

- As per the info esteem given in sequential screen certain means of revolution happens in stepper engine.

For Example

VALUE GIVEN IN SERIAL MONITER	ROTATION
2048	(360) CLK WISE
1024	(180)CLK WISE
512	(90)CLK WISE
-2048	(-360) ANTI CLK WISE
-1024	(-180)ANTI CLK WISE
-512	(-90)ANTI CLK WISE



PROGRAMMING STM32 for Stepper Motor

Like the past instructional exercise, we modified the STM32F103C8 with Arduino IDE through USB port without utilizing FTDI software engineer. To find out about programming STM32 with Arduino IDE follow the connection. We can continue programming it like an Arduino. Complete code is given toward the finish of the venture.

First we require to incorporate the stepper library records #include <Stepper.h> for utilizing stepper capacities.

```
#include <Stepper.h>
```

In this way we characterize no. of steps to finish on revolution, here we utilize 32 in light in case we are utilizing Full-Step (4 Step-grouping) so (360/32 = 11.25 degree). So for one stage, the pole moves 11.25 degree that is stride point. In 4 Step grouping, 4 stages are required for one complete turn.

```
#define STEPS 32
```

We can likewise utilize Half advance mode where there is 8 stage succession (360/64=5.625) stride point.

```
Steps per revolution = 360 / STEP ANGLE
```

As we are setting speed we should take simple incentive from PA4 that is associated with potentiometer. So we should pronounce pin for that

```
const int speedm = PA4
```

At that point we have changed over the simple incentive into computerized by putting away those qualities in factor of whole number sort, after that we require to outline ADC values for setting speed so we utilize the underneath articulation. Become familiar with utilizing ADC with STM32 here.

```
int adc = analogRead(speedm);
int result = map(adc, 0, 4096, 1, 1023);
```

To set speed, we use stepper.setSpeed(result); We have speed scope of (1-1023).

We should make example like beneath to set the pins that are associated with engine. Be cautious in these means as a large portion of them do a mix-up here in this example. They give wrong example and in view of that loops cannot be empowered.

```
Stepper stepper(STEPS, PA0, PA2, PA1, PA3);
```

Underneath proclamation is utilized to get the estimation of steps from sequential screen. For instance we need 2048 qualities for one full revolution (32*64 = 2048) that is 64 will be the apparatus proportion and 32 will be half advance succession for one pivot.

```
rotate = Serial.parseInt();
```

Underneath code is utilized to consider the example and run the engine .If pivot esteem is 1 it calls the capacity stepper one time and one move is finished.

```
stepper.step(rotate);
```

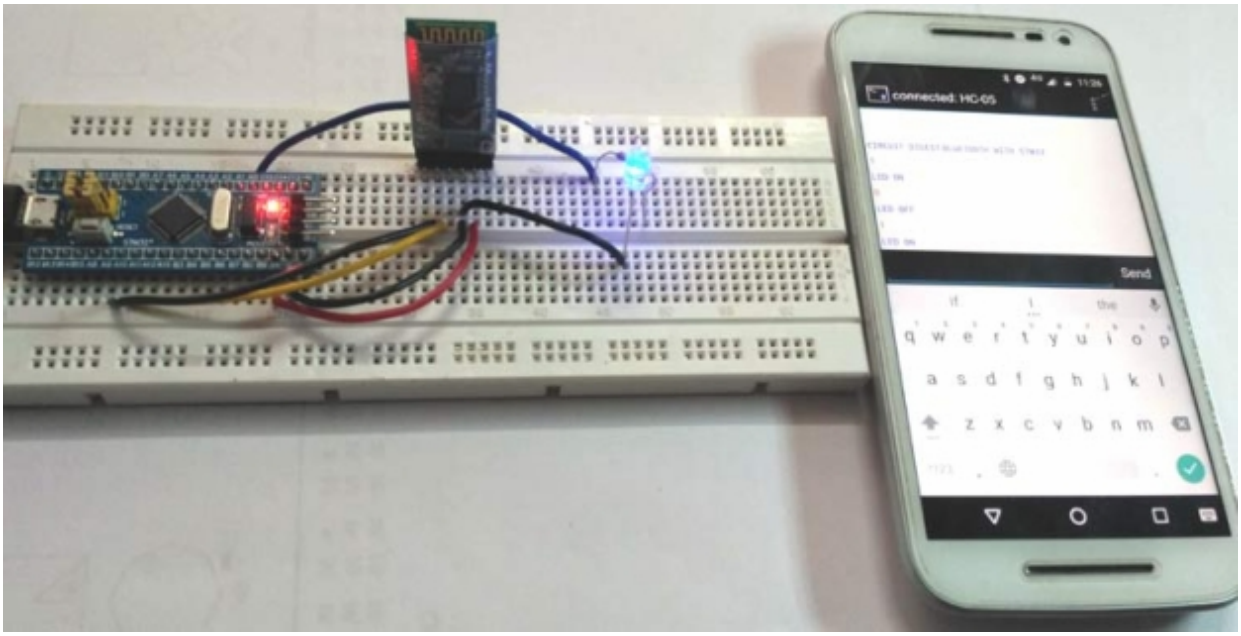
Complete code is given beneath. Likewise check all the stepper engine related ventures here, with interfacing with different microcontrollers

Code

```
//STM32 stepper motor control code
```

```
#include <Stepper.h>                    // Include the Stepper motor header file
#define STEPS 32                        // change this to the number of steps on
your motor
const int speedm = PA4;                 // Pin for input speed
Stepper stepper(STEPS, PA0, PA2, PA1, PA3);  // create an instance of the
stepper class using the steps and pins
int rotate = 0;                         //declare variable rotate with 0 for input
rotation.
void setup()                            //Setup() runs only once
{
  Serial.begin(9600);                   //begins serial communication at
9600baud rate
 pinMode(speedm,INPUT);                 //set pin PA4 as input
}

void loop()                             //loop() runs infinitely
{
 if (Serial.available()>0)              //Checks if serial data is entered or not
in serial monitor
 {
   rotate = Serial.parseInt();          //gets the value for rotation from
serial monitor
  int adc = analogRead(speedm);         //read analog value from pin PA0
   int result = map(adc, 0, 4096, 0, 1023);  //maps the result of ADC from
(0to4096)with (0to1023)
  stepper.setSpeed(result);             //sets the speed of motor
  stepper.step(rotate);                 //makes the motor to rotate
  Serial.println(rotate);               //prints the value you specified to rotate
 }
}
```

7. Interfacing Bluetooth HC-05 with STM32F103C8 Blue Pill: Controlling LED



In this day and age Bluetooth has gotten exceptionally well known and pretty much every gadget like cell phone, PC, and vehicles infotainment framework utilizes Bluetooth for remote correspondence. Bluetooth isn't just used to move information yet moreover to control another Bluetooth gadgets remotely, such as utilizing Bluetooth headset you can hear the melody remotely from your cell phone otherwise can utilize vehicle sound framework to play the tunes from your portable.

Bluetooth is a remote innovation that deals with the recurrence of 2.4GHz. Ordinary Bluetooth signal is in scope of 10 meter range. Bluetooth is most routinely used remote innovation in installed ventures gave that the scope of correspondence is constrained. Bluetooth has included bit of leeway of its low force utilization and minimal effort activity. It is commonly utilized for interfacing microcontrollers with Smart Phones by utilizing Bluetooth applications.

We have seen interfacing of Bluetooth module with different microcontrollers like Arduino, 8051, PIC and so on. Presently in this instructional exercise we will interface a HC-05 Bluetooth module with STM32F103C8 and Turn ON/OFF a LED utilizing Android versatile.

Materials Required

- STM32F103C8
- Bluetooth Module (HC-05)
- Driven
- Android Mobile

- Breadboard

- Interfacing wires

Programming:

- Bluetooth Terminal (Android Application)

Bluetooth Module (HC-05)

It is for the most part utilized Bluetooth module in installed ventures. It is a sequential Bluetooth module that utilizes sequential correspondence having range under 100m and works at 5V (3.3V least). It is often used to associate two microcontrollers remotely and furthermore with cell phone and workstations. As there are numerous android applications are accessible, it is helpful for making remote Bluetooth controlled tasks.

It utilizes USART correspondence and can be linked with microcontrollers having USART correspondence convention.

It has a coordinated radio wire. It has Master/Slave designs that can be changed in AT order mode that is valuable when just a single gadget ought to send the information (ace to slave) like for instance from PC (MASTER) to slave (any MCU). An ace can interface with different gadgets and slave doesn't associate with other association other than ace.

Methods of Operation

It has two modes AT Command Mode and Data Mode.

When Bluetooth is fueled up it enters information mode default. This mode can be used for information moves. To go into AT Command mode during power up we have to press the catch present in module to change the default settings of the module like ace/slave setups.

Pins of Bluetooth Module

- EN pin (ENABLE) - This pin is utilized to set Data Mode or AT Command Mode. Naturally it is in DATA MODE. At the point when catch squeezed during power up it goes to AT order mode.

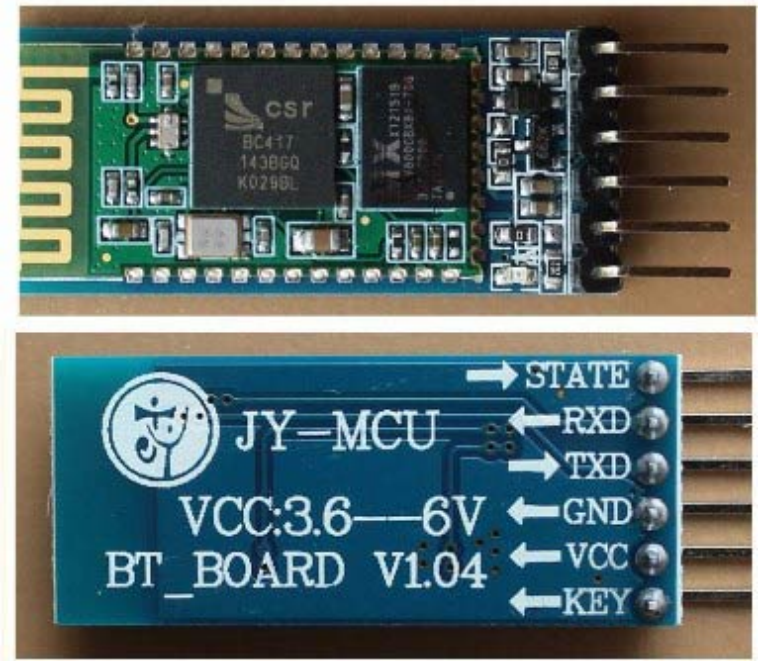
- +5V pin - This is utilized for power supply to the module

- GND pin - This is utilized for ground for module

- TX pin - This pin to associated with RX pin of MCU

- RX pin - This pin associated with TX pin of MCU

- STATE - This pin shows the status of the module, see beneath about signs



Driven Indication

- It has a LED (RED) pointer which gives the condition of the Bluetooth module.

- At the point when the Bluetooth module isn't CONNECTED to any gadget the sign goes low and red drove squints ceaselessly which demonstrate module isn't PAIRED.

- When Bluetooth module is CONNECTED to any gadget the sign goes HIGH and red drove flickers with some defer which demonstrates module is PAIRED.

Check our different ventures to become familiar with Bluetooth module HC-05 with different microcontrollers:

- Bluetooth Controlled Toy Car utilizing Arduino

- Bluetooth Controlled Home Automation System utilizing 8051

- Voice Controlled Lights utilizing Raspberry Pi

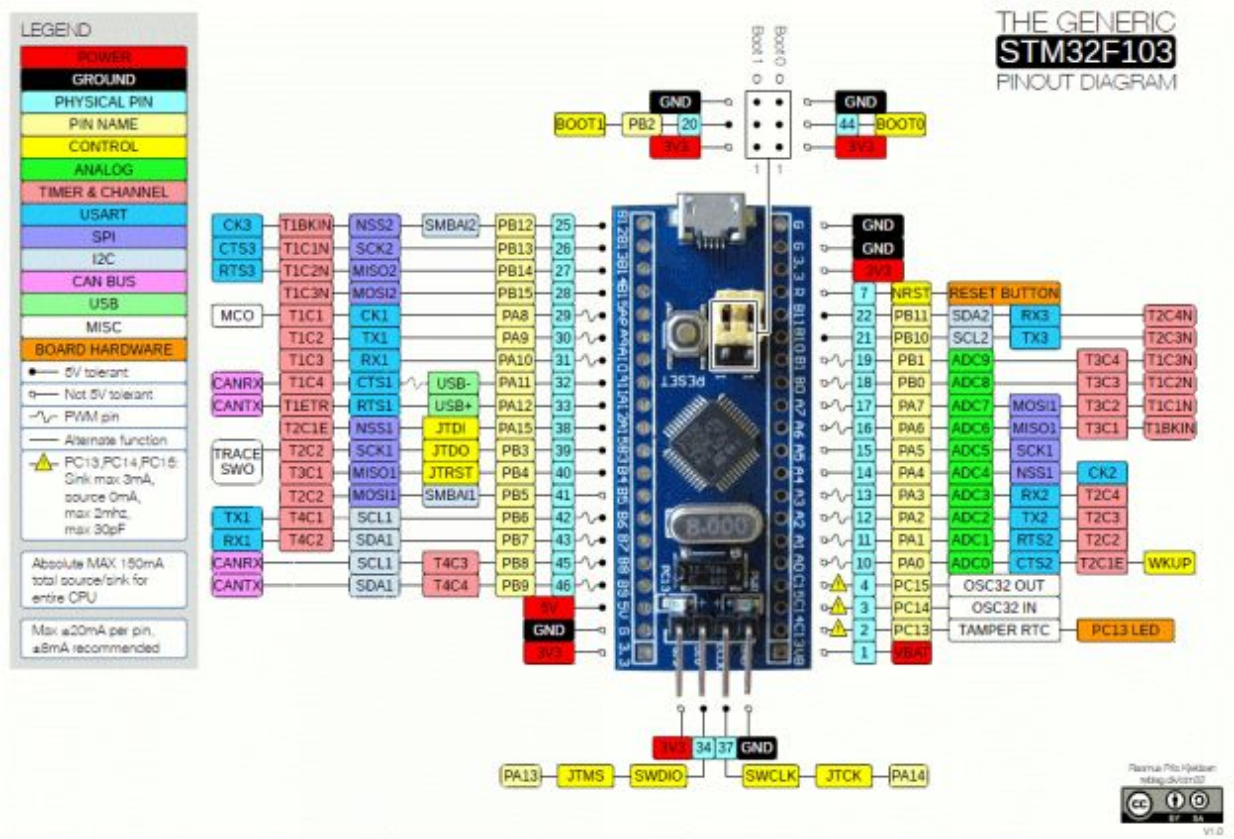
- Advanced mobile phone Controlled FM Radio utilizing Arduino along with Processing

- Cell Phone Controlled Robot Car utilizing G-Sensor and Arduino

- Interfacing Bluetooth Module HC-06 with PIC Microcontroller

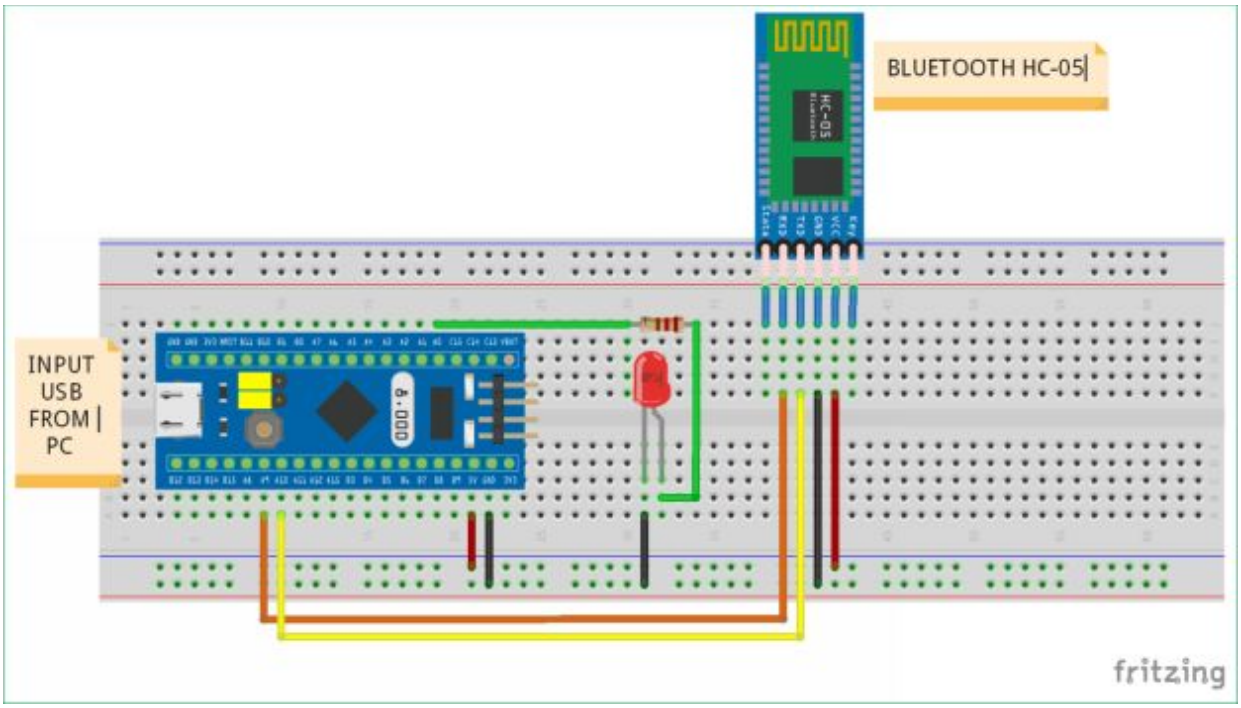
STM32 USART Ports

STM32F103C8 (BLUE PILL) USART sequential correspondence ports are appeared in the pin out picture beneath. These are blue shaded having (PA9-TX1, PA10-RX1, PA2-TX2, PA3-RX2, PB10-TX3, PB11-RX3). It has three such correspondence channels.



Circuit Diagram and Connections

The circuit associations for interfacing Bluetooth Module with STM32 are made like beneath



Association between STM32F103C8 and Bluetooth module (HC-05)

- The TX pin (PA9) of the STM32F103C8 is associated with the RX pin of the Bluetooth module.

- The RX pin (PA10) of the STM32F103C8 is associated with the TX pin of the Bluetooth module.

- VCC (+5V) pin of Bluetooth module is associated with 5V pin of STM32F103C8.

- GND pin of Bluetooth module is associated with GND pin of STM32F103C8.

Different Connections

- The (PA0) pin of STM32 (Blue Pill) is associated with positive pin of LED through an arrangement resistor. The LED is utilized here is blended shading.

- The drove another pin is associated with GND of STM32.

Programming STM32F103C8

Interfacing Bluetooth with STM32 is similar to arduino along with programming in STM32 is similar to Arduino IDE. See this instructional exercise for programming STM32 with USB utilizing Arduino IDE.

As of now told, in this undertaking, we will interface a (HC-05) Bluetooth module with STM32F103C8 and utilize an Android Smart Phone with Bluetooth Terminal android application to kill ON and the LED.

Note: The RX and TX pin must be expelled while transferring the code to STM32F103C8.

Complete code for this venture is given at the end of this instructional exercise.

Coding for this task is so straightforward. Same Arduino codes can be used yet just the pin ought to be changed. Since we have three arrangements of USART pin in STM32F103C8 so we should determine the right pin that we used to interface the Bluetooth module.

1. First we have to name the pins with their particular pin number with int information type as follows

```
const int pinout = PA0;
```

2. Next we need have a variable to store sequential information from the android portable. The information can be a roast or number as follows

```
char inputdata = 0;
```

3. Next in void arrangement(), we should begin sequential correspondence between STM32 Blue Pill and the Bluetooth module by giving baud pace of 9600

```
Serial1.begin(9600);
```

We utilized Serial1 here in light in case we associated HC-05 to TX1 and RX1 of STM32.

We can likewise utilize Serial2 or Serial3 however in like manner pin must be associated.

4. An introduction message is sent as sequential information to the serial1, that is to Bluetooth module HC05. This module further send information to the Bluetooth Terminal application of android portable. So we use underneath articulations

```
Serial1.print("Hello world\n");
Serial1.print("BLUETOOTH WITH STM32\n");
```

5. Next we have to set the pinmode (PA0) as yield , as we associated prompted this pin.so we use

```
pinMode(pinout, OUTPUT);
```

6. Next in the void circle (), we run following information to peruse the sequential information and turn on/off the LED as needs be

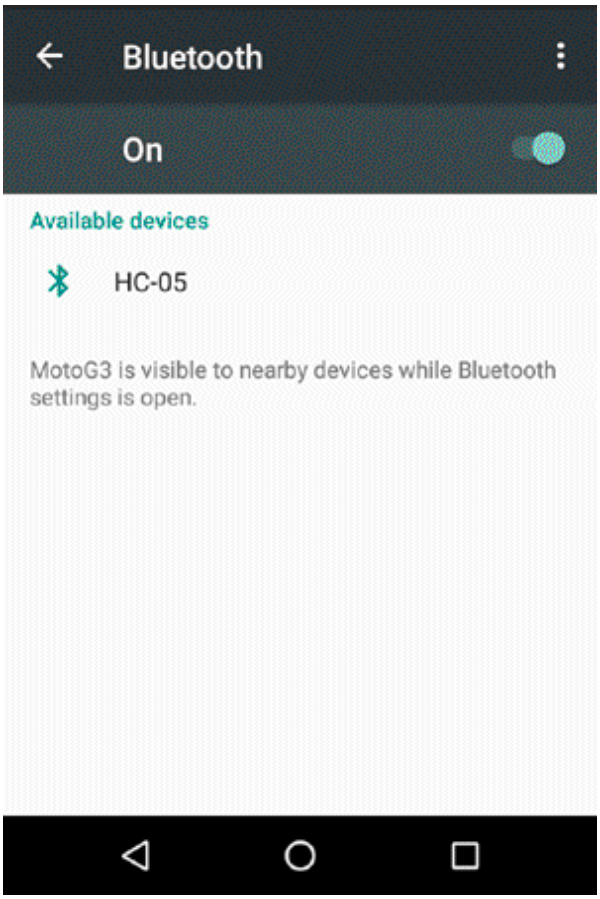
```
void loop()
{
If (Serial1.available() > 0)
{
   inputdata = Serial1.read();
```

```
    if(inputdata == '1')
    {
      digitalWrite(pinout, HIGH);
      Serial1.print("LED ON\n");
    }
    else if(inputdata == '0')
    {
      digitalWrite(pinout, LOW);
      Serial1.print("LED OFF\n");
    }
}
  }
```

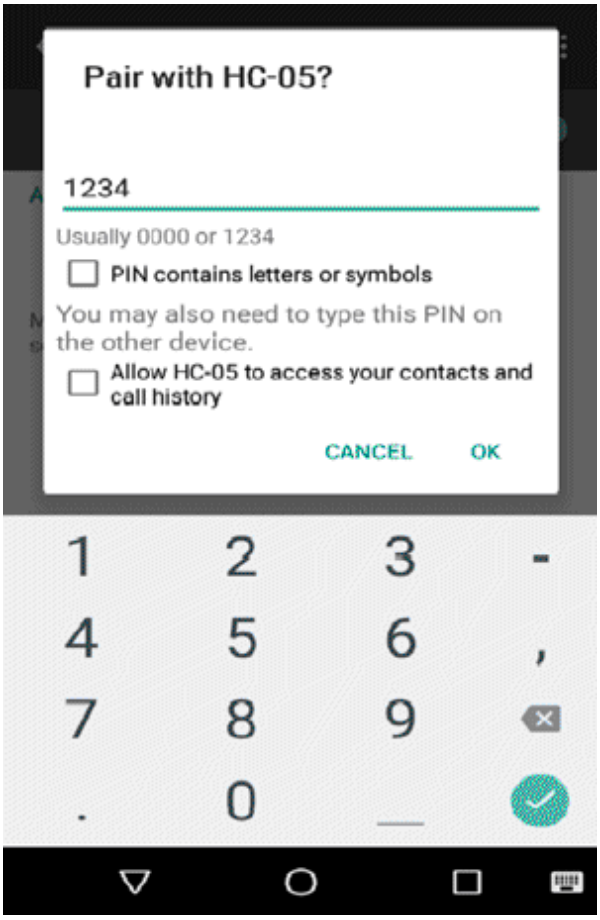
Here we use if explanation in light in case these code possibly executes when Serial1 port has any information gotten from the Bluetooth module that why this announcement is utilized Serial1.available() > 0 . In case it won't get into, it holds up until it begins sequential correspondence. Presently it stores the got information in a variable inputdata = Serial1.read(). At that point it check the worth sent from Bluetooth terminal application. So if esteem is 1, it prints LED ON along with produces the pin (PA0) go HIGH by proclamation digitalWrite(pinout,HIGH) and if the worth is 0, it prints LED OFF along with produces (PA0) pin go LOW.

Steps for Connecting Bluetooth Module with Android Phone

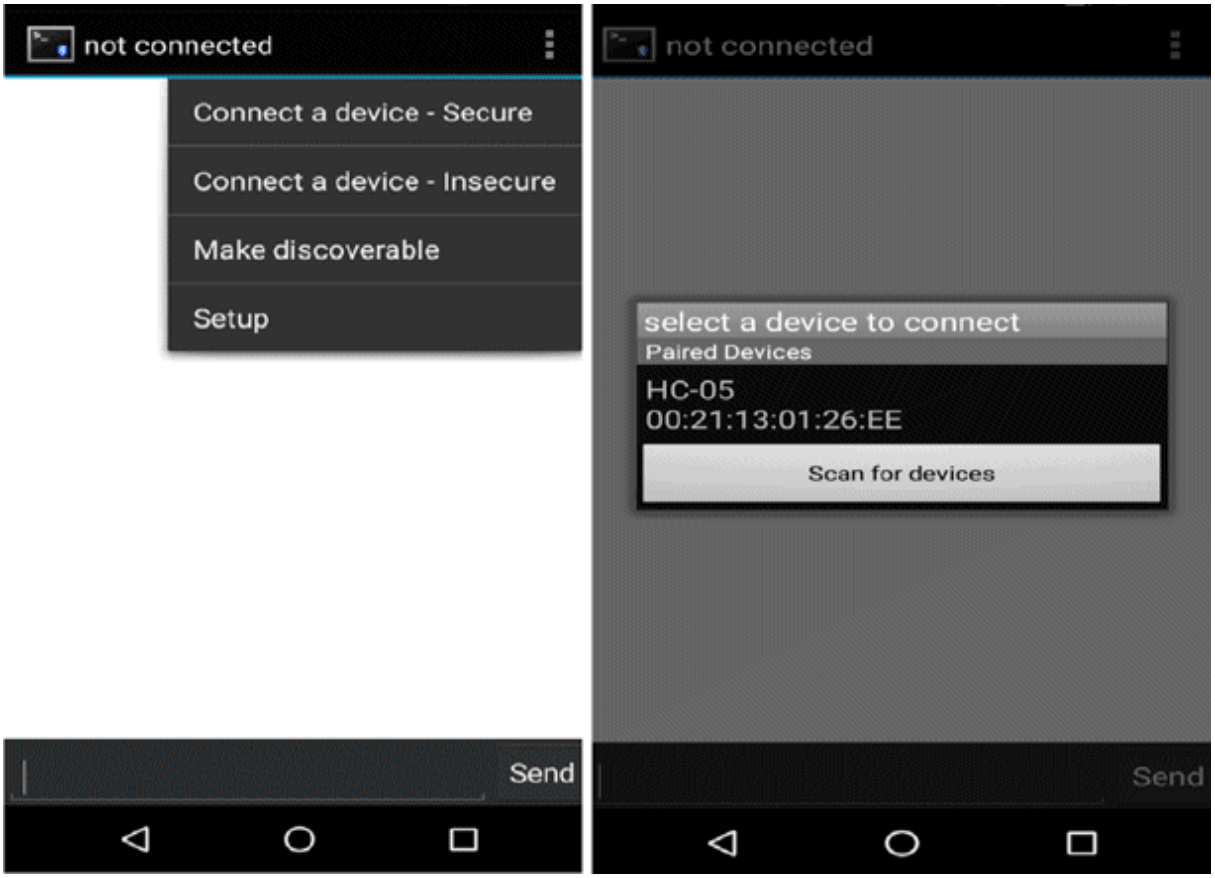
Stage 1:- Open Bluetooth from portable in the wake of transferring code into STM32 from Arduino IDE and offering capacity to circuit. Make sure to evacuate RX and TX pin while UPLOADING code



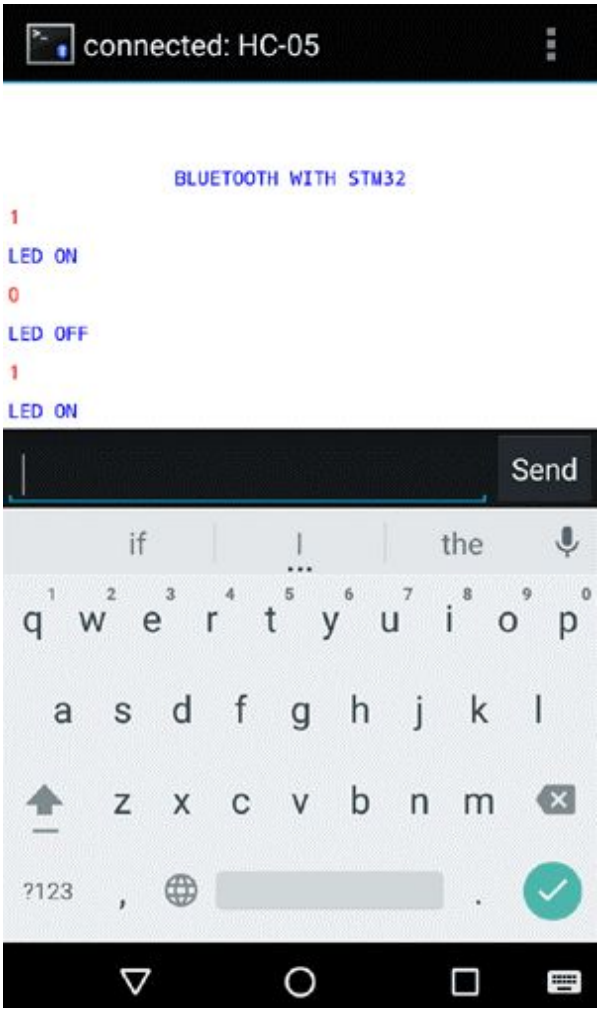
Stage 2:- In accessible gadgets select HC-05 and enter secret word as 1234



Stage 3:- After matching, open Bluetooth Terminal application and select associate a gadget and select HC-05 as demonstrated as follows



Stage 4:- After associating with HC-05 Bluetooth Module, give esteems in the terminal 1 or 0 to kill ON and the LED. You will likewise get a message that LED is On or Off.

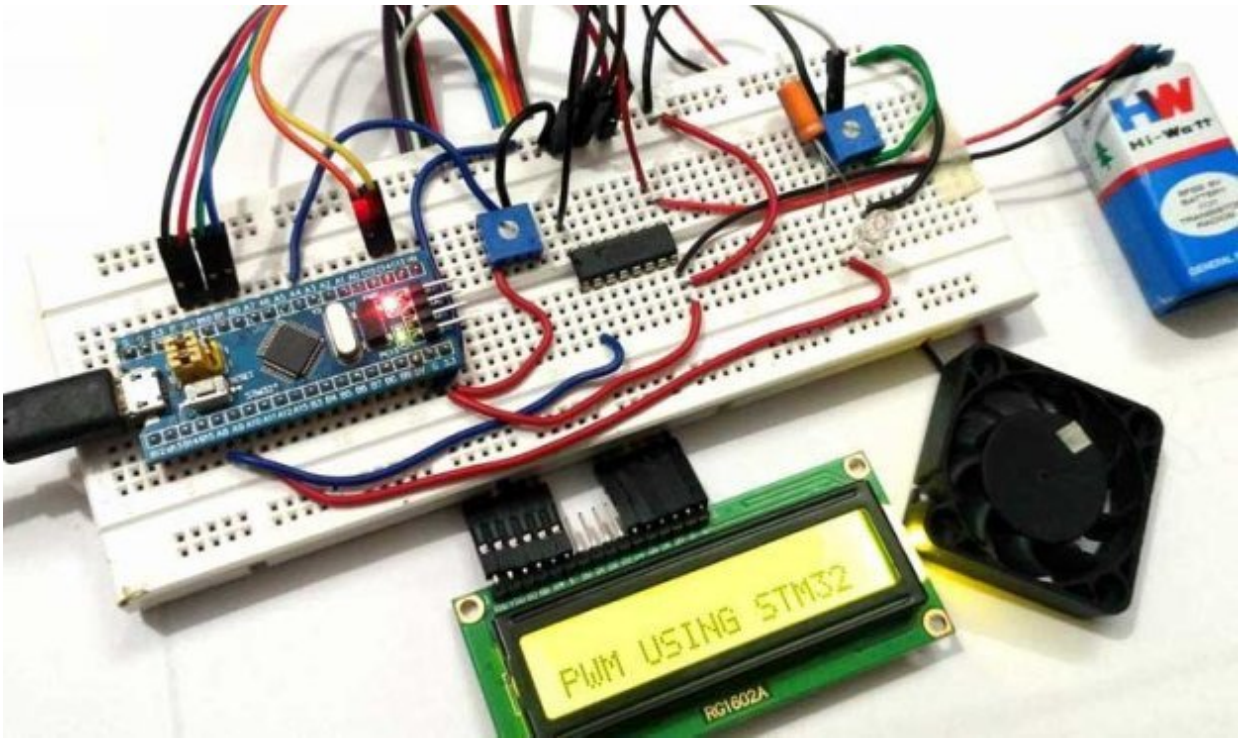


Code

```
//PROGRAM FOR BLUETOOTH INTERFACE WITH STM32F103C8


const int pinout = PA0; // declare pinout with int data type and pin value
char inputdata = 0;  //Variable for storing received data

void setup()
{
   Serial1.begin(9600);                  //Sets the baud rate for bluetooth pins
   Serial1.print("Hello world\n");
   Serial1.print("BLUETOOTH WITH STM32\n");
    pinMode(pinout, OUTPUT);                  //Sets digital pin PA0 as output
```

```
pin for led
}

void loop()
{
  if(Serial1.available() > 0)      // Send data only when you receive data:
   {
      inputdata = Serial1.read();          //Read the incoming data & store into
data



      if(inputdata == '1')
     {
       digitalWrite(pinout, HIGH);
       Serial1.print("LED ON\n");
     }


      else if(inputdata == '0')
    {
      digitalWrite(pinout, LOW);
      Serial1.print("LED OFF\n");
    }
  }
}
```

8. Heartbeat width Modulation (PWM) in STM32F103C8: Controlling Speed of DC Fan



In past article we have seen about ADC change utilizing STM32. In this instructional exercise, we will find out about PWM (Pulse Width Modulation) in STM32 and how might we control splendor of LED or speed of DC fan utilizing PWM method.

We realize that there are 2 kinds of sign: Analog and Digital. Simple signs have voltages like (3V, 1V...etc) and computerized signals have (1' and 0's). Sensors yields are of simple signs and these simple signs are changed over into advanced utilizing ADC, in light of the fact that microcontrollers just comprehend computerized. In the wake of handling those ADC esteems, again the yield should be changed over into simple structure to drive the simple gadgets. For that we utilize certain techniques like Pulse Width Modulation, Digital to Analog converters and so forth.

What is PWM (Pulse with Modulation)?

PWM is an approach to control the simple gadgets utilizing advanced worth like controlling engine's speed, brilliance of a drove along with etc. We realize that engine and drove takes a shot at simple sign. Be that as it may, the PWM doesn't give unadulterated simple yield, PWM seems as though simple sign made by short heartbeats, which is given by obligation cycle.

Obligation pattern of the Pulse Width Modulation

The level of time wherein the Pulse Width Modulation signal stays HIGH (on schedule) is called as obligation cycle. In case the sign is consistently ON it is in 100% obligation cycle and in case it is constantly off it is 0% obligation cycle.

Duty Cycle =Turn ON time/ (Turn ON time + Turn OFF time)

PWM in STM32

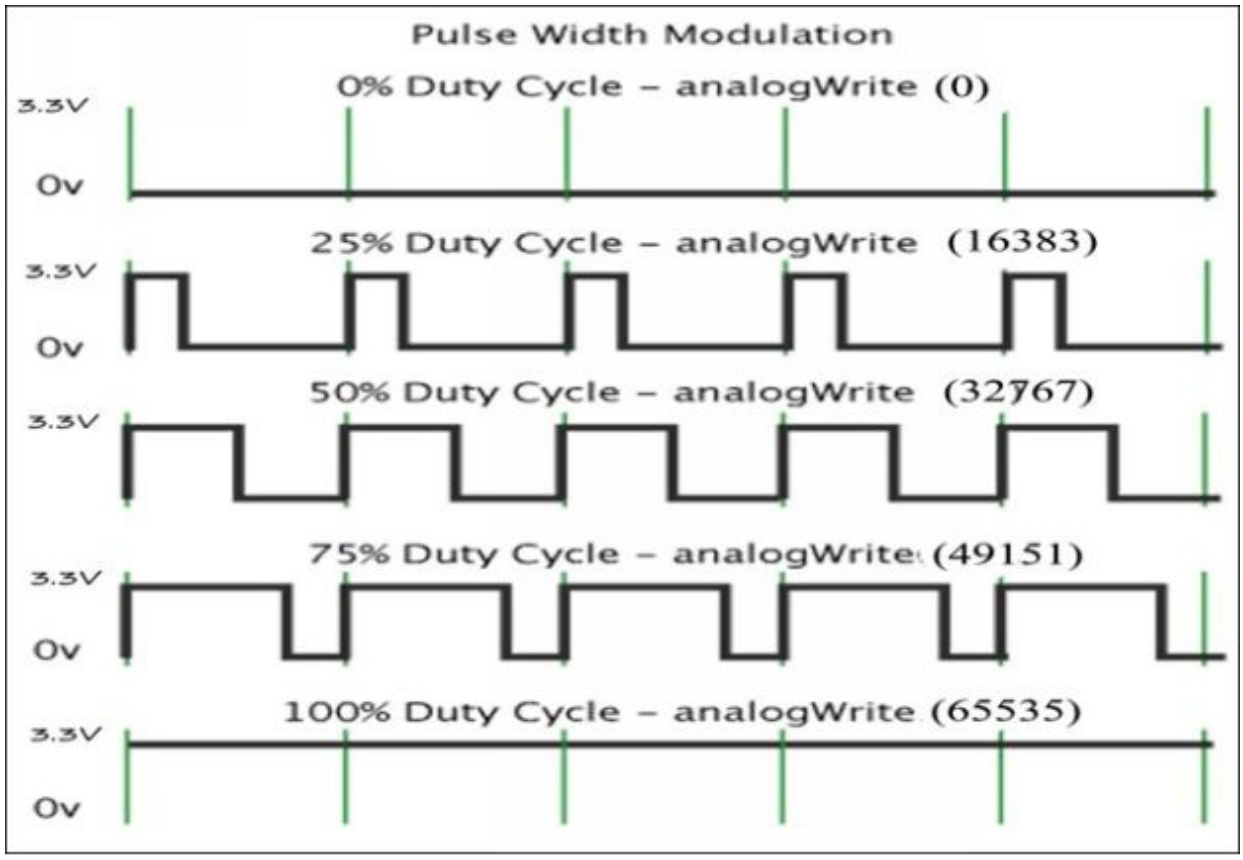
STM32F103C8 has 15 PWM pins and 10 ADC pins. There are 7 clocks and each PWM yield is given by a channel associated with 4 clocks. It has 16-piece PWM goals (216), that is counters and factors can be as extensive

as 65535. With a 72MHz clock rate, a PWM yield can have most extreme time of around one millisecond.

- So estimation of 65535 gives FULL BRIGHTNESS of LED AND FULL SPEED of DC Fan (100% Duty Cycle)

- In like manner estimation of 32767 gives HALF BRIGHTNESS of LED AND HALF SPEED of DC Fan (half Duty Cycle)

- What's more, estimation of 13107 gives (20%) BRIGHTNESS AND (20%) SPEED (20% Duty Cycle)



In this instructional exercise, we are utilizing potentiometer and STM32 to fluctuate the brilliance of LED and speed of a DC fan by PWM procedure. A 16x2 LCD is utilized to show ADC esteem (0-4095) and the changed variable (PWM esteem) that is yield (0-65535).

Here are not many PWM models with other Microcontroller:

- Creating PWM utilizing PIC Microcontroller with MPLAB along with XC8
- Servo Motor Control with Raspberry Pi
- Arduino Based Light Emitting Diode Dimmer utilizing PWM
- Heartbeat width Modulation (PWM) utilizing MSP430G2

Check all the PWM related activities here.

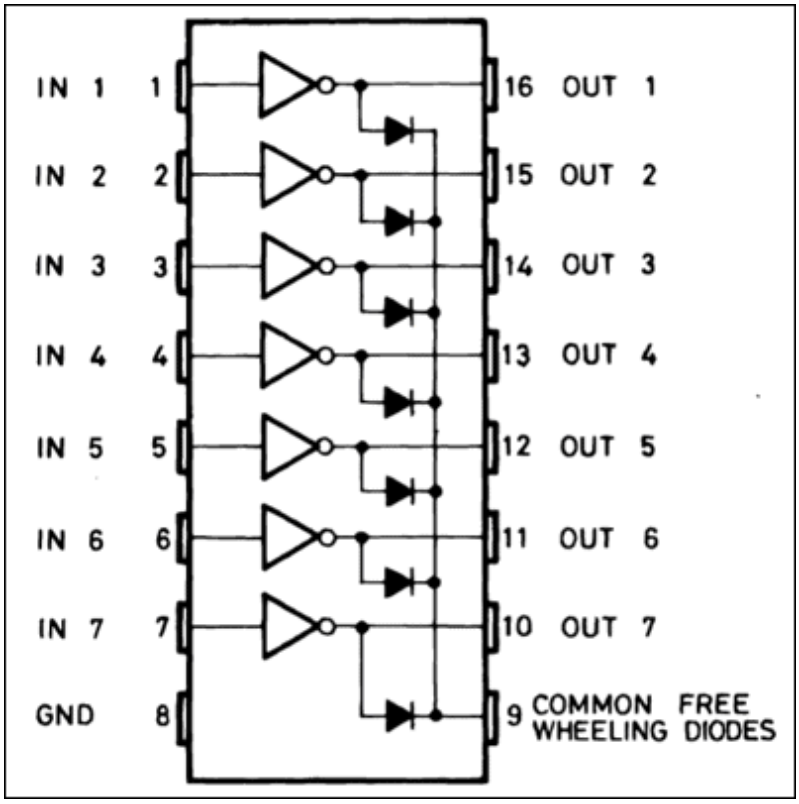
Parts Required

- STM32F103C8
- DC fan
- ULN2003 Motor Driver IC
- Driven (RED)

- LCD (16x2)
- Potentiometer
- Breadboard
- Battery 9V
- Jumper Wires

DC Fan: The DC fan utilized here is BLDC fan from an old PC .It requires an outside inventory so we are utilizing a 9V dc battery.

ULN2003 Motor Driver IC: It is utilized to drive the engine one way as the engine is unidirectional and furthermore outer force is required for fan. Study ULN2003 based Motor Driver Circuit here. The following is the pic outline of ULN2003:

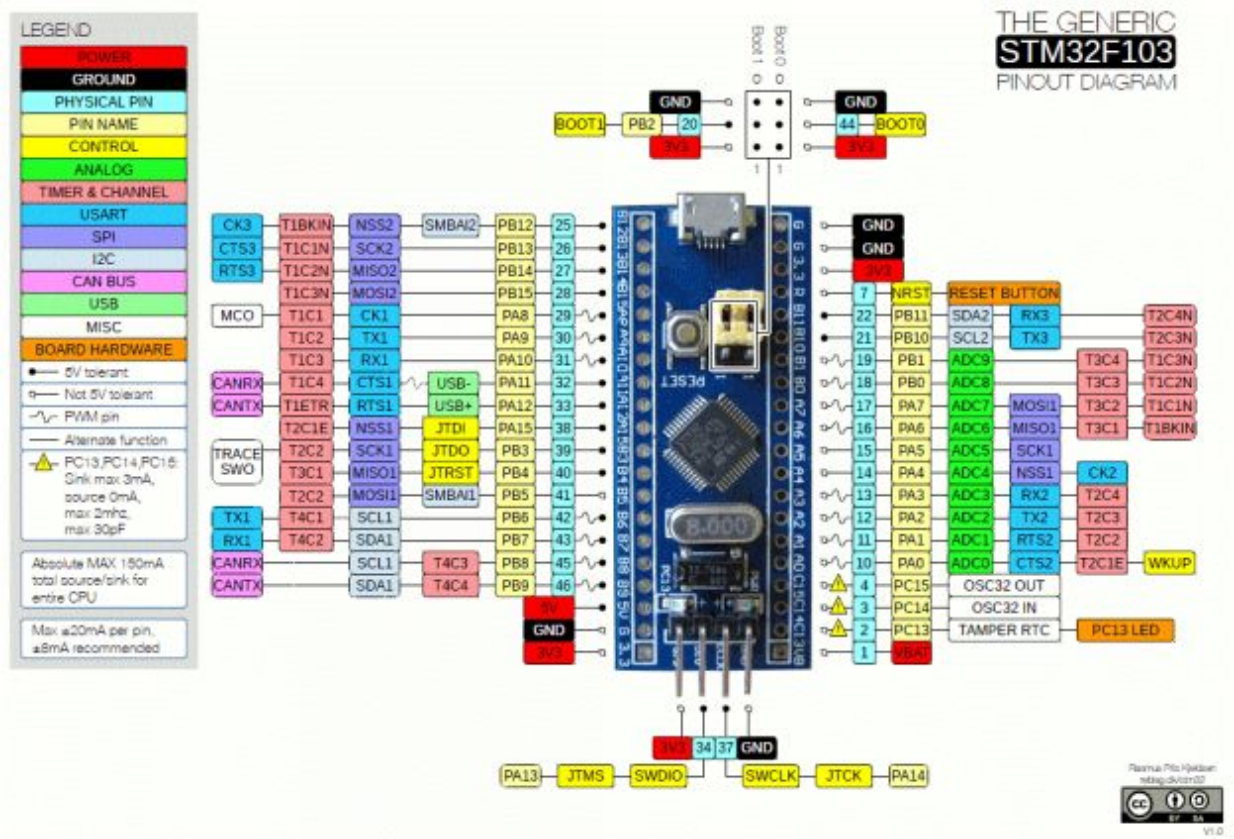


Pins (IN1 to IN7) are input sticks and (OUT 1 to OUT 7) are relating yield pins. COM is given Positive source voltage required for yield gadgets.

Driven: RED hued drove is utilized which transmits RED light. Any hues can be used.

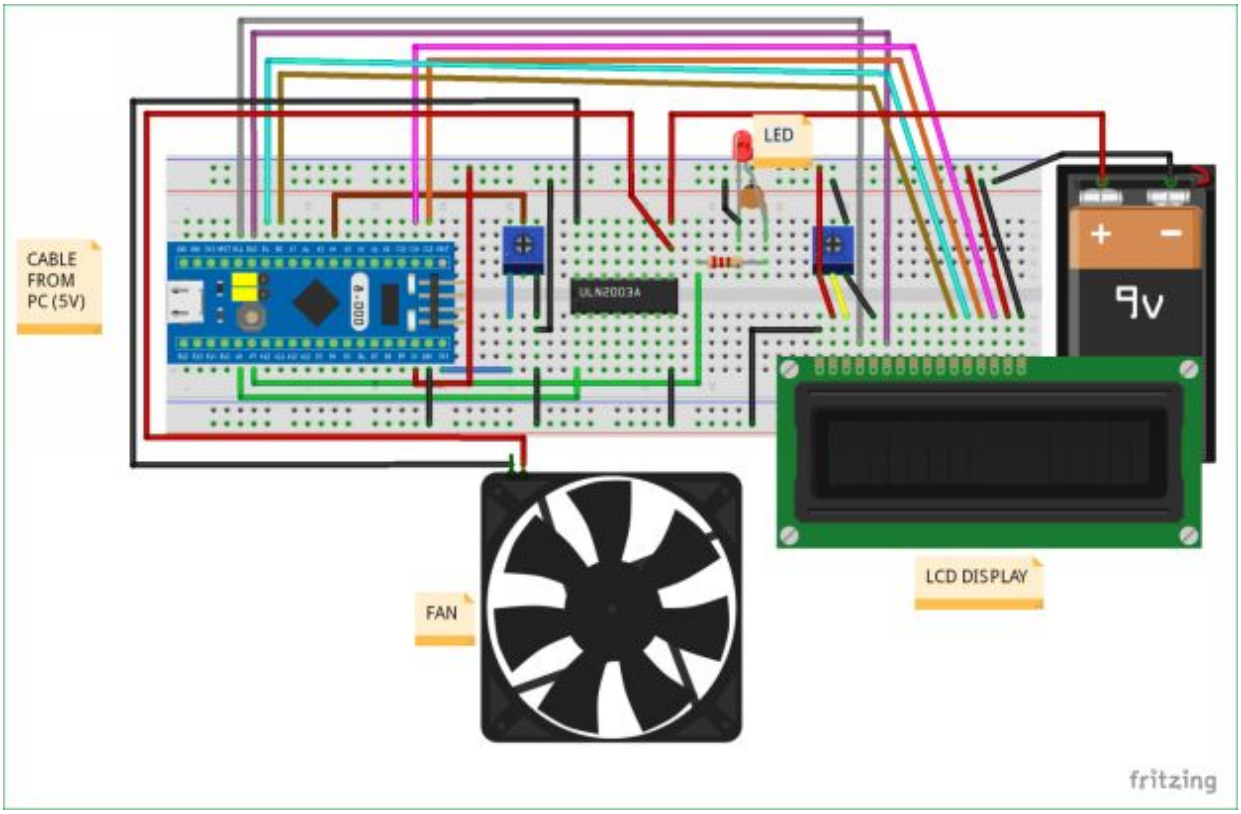
Potentiometers: Two potentiometers are utilized one is for voltage divider for simple contribution to ADC and another is for controlling brilliance of drove.

Pin Details of STM32



As should be obvious the PWM pins are shown in wave group (~), there are 15 such pins, ADC pins are representd in green shading, 10 ADC pins are there which are utilized for simple sources of info.

Circuit Diagram and Connections



Associations of STM32 with different parts are clarified as underneath:

STM32 with Analog Input (ADC)

The potentiometer present at left half of circuit is utilized as voltage controller that manages voltage from the 3.3V pin. The yield from the potentiometer for example focus pin of potentiometer is associated with the ADC pin (PA4) of STM32.

STM32 with LED

The STM32 PWM yield pin (PA9) is associated with the positive pin of LED through an arrangement resistor and a capacitor.

Driven with Resistor and Capacitor

A resistor in arrangement and a capacitor in equal are associated with LED to create right Analog wave from PWM yield as simple yield isn't in unadulterated from when produced straightforwardly from PWM pin.

STM32 with ULN2003 and ULN2003 with Fan

STM32 PWM yield pin (PA8) is associated with the Input pin (IN1) of ULN2003 IC and the comparing yield pin (OUT1) of ULN2003 is associated with negative wire of the DC FAN.

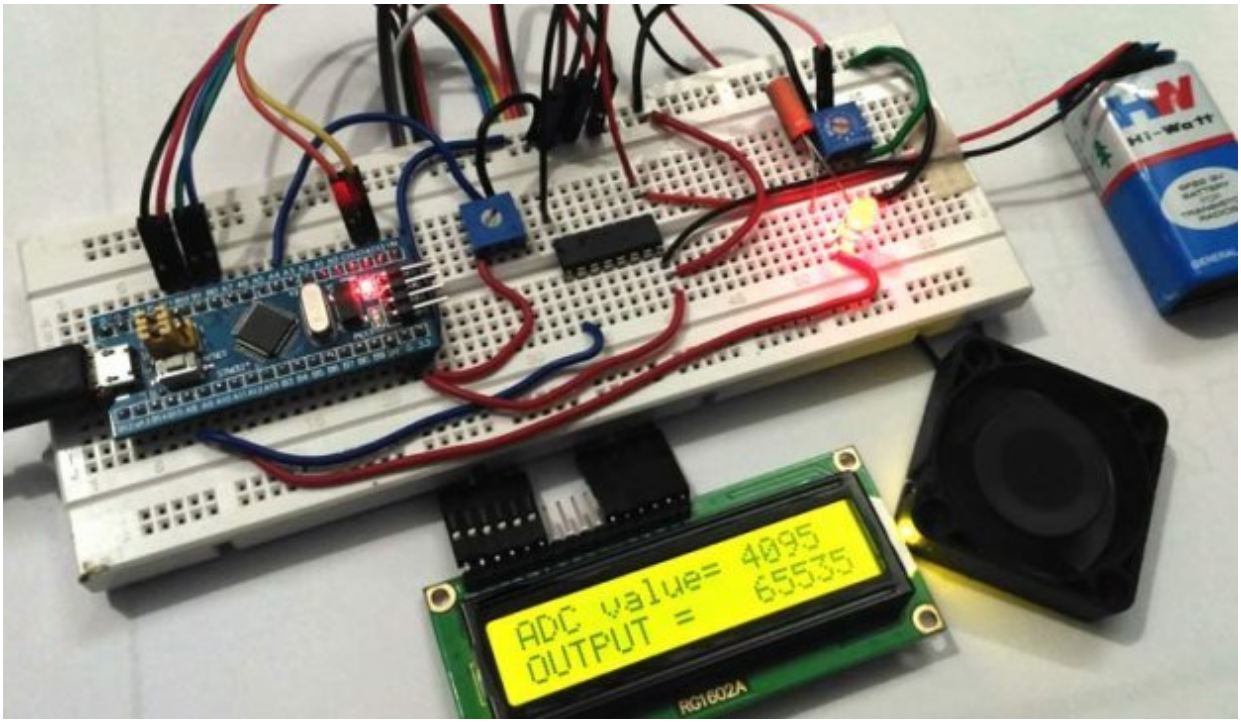
Positive pin of the DC fan is associated with the COM pin of the ULN2003 IC and the outer battery (9V DC) is likewise associated with the equivalent COM pin of the ULN2003 IC. GND pin of ULN2003 is associated with GND pin of STM32 and battery negative is associated with same GND pin.

STM32 with LCD (16x2)

LCD Pin No	LCD Pin Name	STM32 Pin Name
1	Ground (Gnd)	Ground (G)
2	VCC	5V
3	VEE	Pin from Centre of Potentiometer
4	Register Select (RS)	PB11
5	Read/Write (RW)	Ground (G)
6	Enable (EN)	PB10
7	Data Bit 0 (DB0)	No Connection (NC)
8	Data Bit 1 (DB1)	No Connection (NC)
9	Data Bit 2 (DB2)	No Connection (NC)
10	Data Bit 3 (DB3)	No Connection (NC)
11	Data Bit 4 (DB4)	PB0
12	Data Bit 5 (DB5)	PB1
13	Data Bit 6 (DB6)	PC13
14	Data Bit 7 (DB7)	PC14

15	LED Positive	5V
16	LED Negative	Ground (G)

A potentiometer on the correct side is utilized to manage the difference of the Liquid Crystal Display show. The above table shows the association among LCD and STM32.



Programming STM32

Like the past instructional exercise, we modified the STM32F103C8 with Arduino IDE through USB port without utilizing FTDI software engineer. To find out about programming STM32 with Arduino IDE follow the connection. We can continue programming as like in Arduino. Complete code is given toward the end.

In this coding we are going to take an information simple incentive from ADC pin (PA4) which is associated with focus pin of left potentiometer and afterward convert the Analog worth (0-3.3V) into advanced or whole number arrangement (0-4095). This advanced worth is additionally given as

PWM yield to control LED brilliance and speed of DC fan. A 16x2 LCD is utilized to show ADC and mapped esteem (PWM yield esteem).

First we have to incorporate LCD header document, announce LCD sticks and instate them utilizing beneath code. Become familiar with interfacing LCD with STM32 here.

```
#include <LiquidCrystal.h>    // include the LCD library
const int rs = PB11, en = PB10, d4 = PB0, d5 = PB1, d6 = PC13, d7 = PC14;  //mention the pin names to with LCD is connected to
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);  //Initialize the LCD
```

Next announce and characterize the pin names utilizing the pin of STM32

```
const int analoginput   = PA4;  // Input from potentiometer

const int led          = PA9;  // LED output
const int fan          = PA8;  // fan output
```

Presently inside the arrangement(), we have to show a few messages and clear them following couple of moments and determine the INPUT pin and PWM yield pins

```
lcd.begin(16,2);         //Getting LCD ready
lcd.clear();         //Clears LCD
lcd.setCursor(0,0);         //Sets cursor at row0 and column0
lcd.print("Hello world");         //Displays Hello world
lcd.setCursor(0,1);         //Sets Cursor at column0 and row1
lcd.print("PWM USING STM32");   //Displays PWM using STM32
delay(2000);         // Delay Time
lcd.clear();         // Clears LCD
pinMode(analoginput, INPUT);          // set pin mode analoginput as INPUT
pinMode(led, PWM);         // set pin mode led as PWM output
pinMode(fan, PWM);         // set pin mode fan as PWM output
```

The Analog information pin (PA4) is set as INPUT by pinMode(analoginput, INPUT), LED pin is set as PWM yield by pinMode(led, PWM) and fan pin is set as PWM yield by pinMode(fan, PWM). Here the PWM yield pins are associated with LED (PA9) and Fan (PA8).

Next in void circle() work, we read the Analog sign from the ADC pin (PA4) and store it in a whole number variable that changes over simple voltage into advanced number qualities (0-4095) by utilizing beneath code int valueadc = analogRead(analoginput);

Significant thing to note here is PWM pins that is channels of STM32 has 16-Bit goals (0-65535) so we have to outline with simple qualities utilizing map work like underneath

```
int result = map(valueadc, 0, 4095, 0, 65535).
```

On the off chance that mapping isn't utilized we won't get max throttle of fan or full brilliance of LED by fluctuating the potentiometer.

At that point we compose the PWM yield to the LED by utilizing pwmWrite(led, result) and PWM yield to fan by utilizing pwmWrite(fan, result) capacities.

Finally we show the Analog information esteem (ADC esteem) and the yield esteems (PWM values) on LCD show by utilizing following orders

```
lcd.setCursor(0,0);          //Sets cursor at row0 and column0
lcd.print("ADC value= ");          // prints the words “”
lcd.print(valueadc);          //displays valueadc
lcd.setCursor(0,1);          //Sets Cursor at column0 and row1
lcd.print("Output =   ");          //prints the words in ""
lcd.print(result);          //displays value result
```

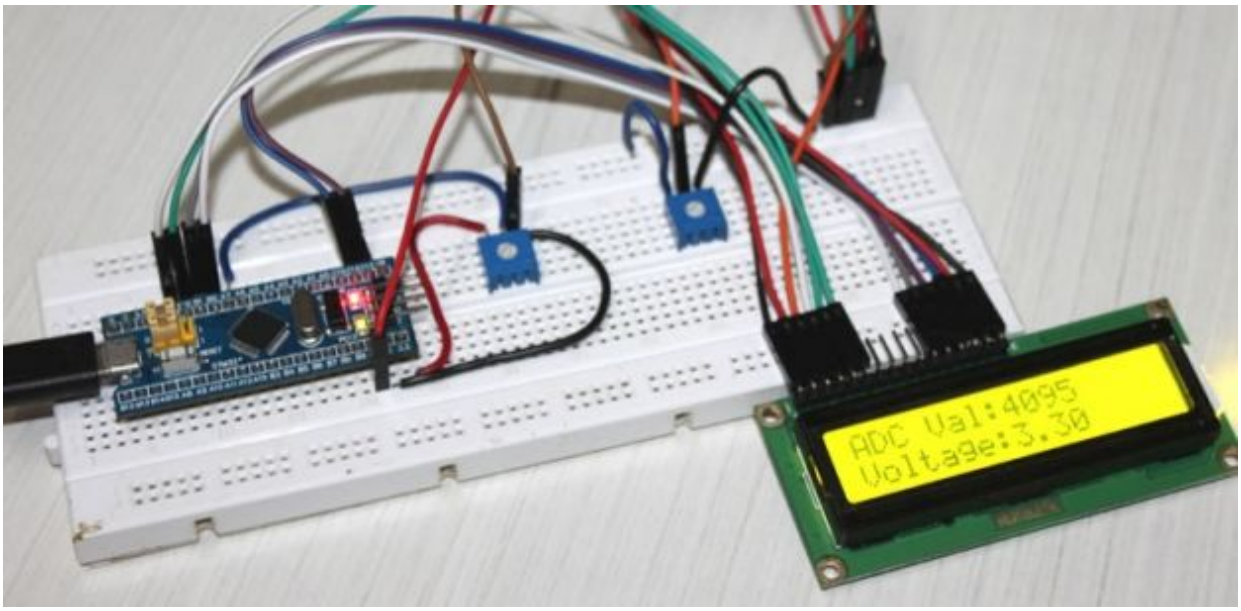
Complete Code is given underneath.

Code

```
#include <LiquidCrystal.h>     // include the LCD library
const int rs = PB11, en = PB10, d4 = PB0, d5 = PB1, d6 = PC13, d7 = PC14;  //mention the pin names to with LCD is connected to
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);  //Initialize the LCD

const int analoginput   = PA4;  // Input from potentiometer
const int led           = PA9;  // LED output
const int fan           = PA8;  // fan output

void setup()
{
   lcd.begin(16,2);     //Getting LCD ready
   lcd.clear();  //Clears LCD
   lcd.setCursor(0,0);  //Sets cursor at row0 and column0
   lcd.print("Hello world");   //Displays Hello world
   lcd.setCursor(0,1);         //Sets Cursor at column0 and row1
   lcd.print("PWM USING STM32");   //Displays PWM using STM32
   delay(2000);       // Delay yime
   lcd.clear();       // Clears LCD
   pinMode(analoginput, INPUT);   // set pin mode analoginput as INPUT
   pinMode(led, PWM);           // set pin mode led as PWM output
   pinMode(fan, PWM);           // set pin mode fan as PWM output
}


void loop()

  {
    int valueadc = analogRead(analoginput);      //gets analog value from pot and store in variable,converts to digital
     int result = map(valueadc, 0, 4095, 0, 65535); //maps the (0to4095 into 0to65535) and stores in result variable
   pwmWrite(led, result);          //puts resultin PWM form
   pwmWrite(fan, result);          //puts resultin PWM form
   lcd.setCursor(0,0);            //Sets cursor at row0 and column0
   lcd.print("ADC value= ");          // prints the words in ""
   lcd.print(valueadc);          //displays value
   lcd.setCursor(0,1);            //Sets Cursor at column0 and row1
```

```
  lcd.print("Output =   ");          //prints the words in ""
  lcd.print(result);           //displays value result
}
```

9. The more effective method to utilize ADC in STM32F103C8 - Measuring Analog Voltage



One regular component that is utilized in pretty much every implanted application is the ADC module (Analog to Digital Converter). These Analog

to computerized Converters can peruse voltage from simple sensors like Temperature sensor, Tilt sensor, Current sensor, Flex sensor and considerably more. So in this instructional exercise we will figure out how to utilize ADC in STM32F103C8 to peruse Analog voltages utilizing the Energia IDE. We will interface a little potentiometer to STM32 Blue Pill board along with supply a changing voltage to an Analog pin, read the voltage along with show it on the 16x2 Liquid Crystal Display screen.

Looking at ADC in Arduino and STM32F103C8

In Arduino board, it contains a 6 channel (8 channels on the Mini and Nano, 16 on the Mega), 10-piece ADC with an information voltage scope of 0V–5V. This implies it will delineate voltages somewhere in the range of 0 and 5 volts into whole number qualities somewhere in the range of 0 and 1023. Presently on account of STM32F103C8 we have 10 channels, 12-Bit ADC with an info go 0V - 3.3V. It will outline voltages somewhere in the range of 0 and 3.3 volts into whole number qualities somewhere in the range of 0 and 4095.

ADC in STM32

The ADC inserted in STM32 microcontrollers utilizes the SAR (progressive estimate register) rule, by which the transformation is acted in a few stages. The quantity of change steps identical to the quantity of bits in the ADC converter. Each progression is driven by the ADC clock. Each ADC clock produces the slightest bit from result to yield. The ADC inner structure depends on the exchanged capacitor method. In the event that you are new to STM32, at that point checkout our Getting started with STM32 instructional exercise.

12-piece Resolution

This ADC is a 10 channel 12 - bit ADC. Here the term 10 channel infers that there are ten ADC pins utilizing which we can gauge simple voltage. The term 12-piece suggests the goals of the ADC. 12-piece implies 2 to the intensity of ten (212) which is 4096. This is the quantity of test ventures for our ADC, so the scope of our ADC esteems will be from 0 to 4095. The

worth will increment from 0 to 4095 dependent on the estimation of voltage per step, which can be determined by recipe

VOLTAGE / STEP = REFERENCE VOLTAGE / 4096 = (3.3/4096= 8.056mV) per unit.

How an Analog Signal is changed over into Digital Format

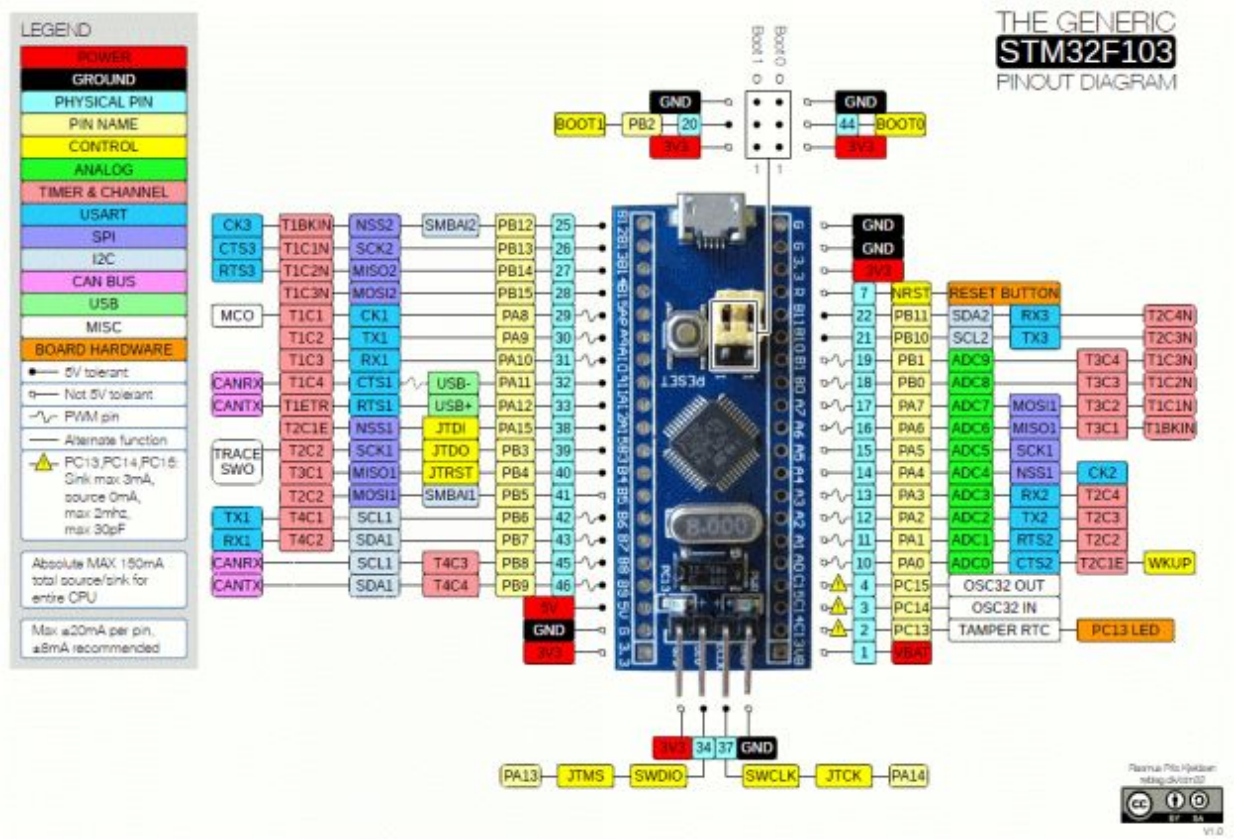
As PCs store and procedure just paired/computerized values (1's and 0's). So Analog signs like sensor's yield in volts must be changed over into advanced qualities for preparing and the transformation should be precise .When an information simple voltage is given to STM32 at its Analog data sources, the simple worth is perused and put away in a whole number variable. That put away Analog value(0-3.3V) is changed over into numbers esteems (0-4096) utilizing the recipe underneath:

INPUT VOLTAGE = (ADC Value / ADC Resolution) * Reference Voltage

Goals = 4096

Reference = 3.3V

ADC Pins in STM32F103C8T6



There are ten Analog Pins in STM32 from PA0 to PB1.

Additionally check how to utilize ADC in different Microcontrollers:

- How to Use ADC in Arduino Uno?
- Interfacing ADC0808 with 8051 Microcontroller
- Utilizing ADC Module of PIC Microcontroller
- Raspberry Pi ADC Tutorial

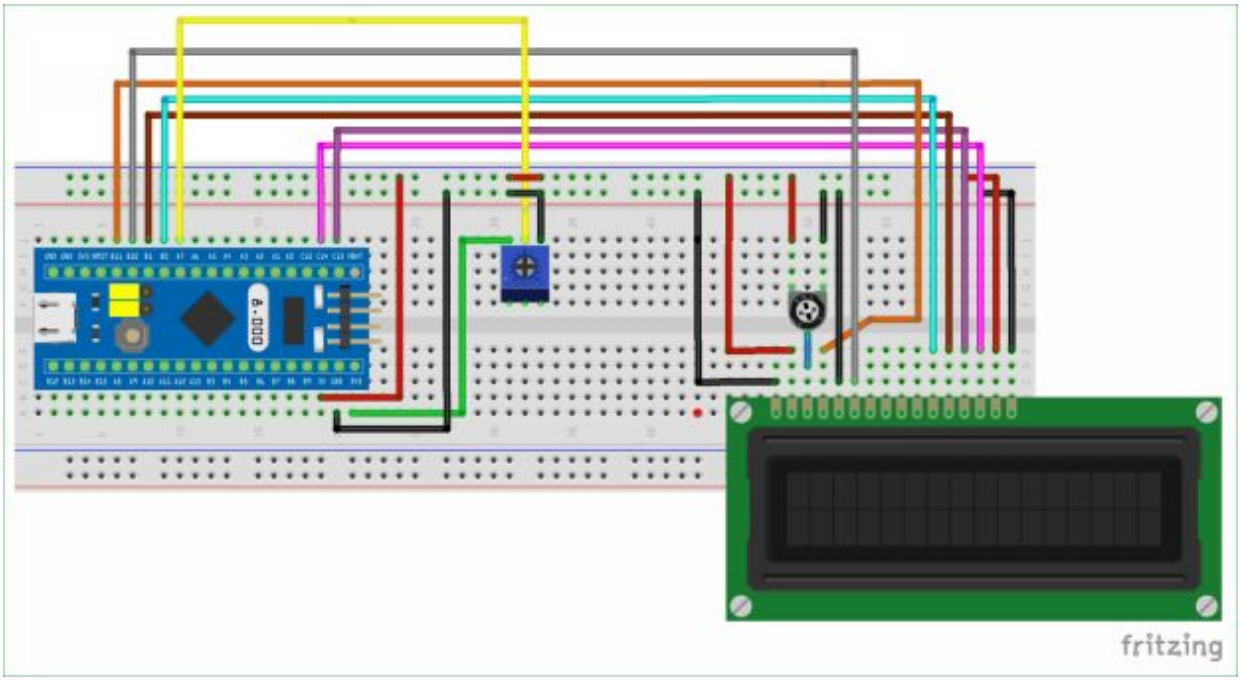
- The most effective method to utilize ADC in MSP430G2 - Measuring Analog Voltage

Parts Required

- STM32F103C8
- LCD 16*2
- Potentiometer 100k
- Breadboard
- Interfacing wires

Circuit Diagram and Explanations

The circuit outline to interface 16*2 LCD and Analog Input to a STM32F103C8T6 board is demonstrated as follows.

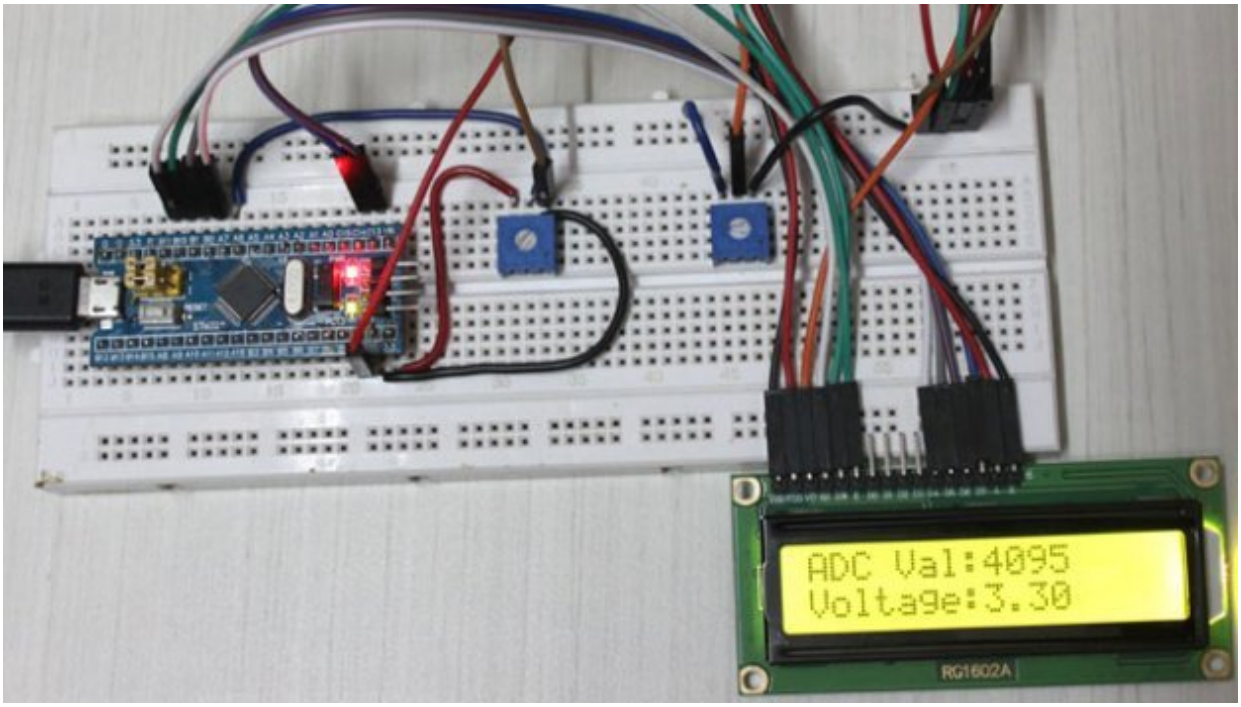


The associations which are accomplished for LCD are given underneath:

LCD Pin No	LCD Pin Name	STM32 Pin Name
1	Ground (Gnd)	Ground (G)
2	VCC	5V
3	VEE	Pin from Centre of Potentiometer
4	Register Select (RS)	PB11
5	Read/Write (RW)	Ground (G)
6	Enable (EN)	PB10
7	Data Bit 0 (DB0)	No Connection (NC)
8	Data Bit 1 (DB1)	No Connection (NC)
9	Data Bit 2 (DB2)	No Connection (NC)
10	Data Bit 3 (DB3)	No Connection (NC)

11	Data Bit 4 (DB4)	PB0
12	Data Bit 5 (DB5)	PB1
13	Data Bit 6 (DB6)	PC13
14	Data Bit 7 (DB7)	PC14
15	LED Positive	5V
16	LED Negative	Ground (G)

The associations are made by the above given table. There are two Potentiometers present in the circuit, initial one is utilized for voltage divider which can be utilized to change voltage and give simple contribution to STM32. Left pin of this potentiometer gets input positive voltage from STM32 (3.3V) and right pin is associated with ground, focus pin of potentiometer is associated with simple information pin (PA7) of STM32. The other potentiometer is utilized to fluctuate the complexity of the Liquid Crystal Display show. The force hotspot for STM32 is given by methods for USB power supply from a PC or Laptop.



Programming STM32 for perusing ADC values

In our past instructional exercise, we found out about Programming STM32F103C8T6 Board utilizing USB Port . So we needn't bother with a FTDI developer now. Just interface it to PC by means of USB port of STM32 and begin programming with ARDUINO IDE. Programming your STM32 in ARDUINO IDE to peruse simple voltage is basic. It is same like arduino board. There is no need of changing the jumper pins of STM32.

In this program will peruse the simple esteem and figure the voltage with that worth and afterward show both, simple and computerized values, on the LCD screen.

First characterize out LCD pins. These characterize to which pin of STM32 the LCD pins are associated. You can adjust according to your necessities.

```
const int rs = PB11, en = PB10, d4 = PB0, d5 = PB1, d6 = PC13, d7 = PC14; //mention the pin names to with LCD is connected to
```

Next, we incorporate the header record for the LCD show. This calls the library which contains the code for how the STM32 ought to speak with the LCD. Likewise ensure the capacity Liquid Crystal is called with the pin names that we simply characterized previously.

```
#include <LiquidCrystal.h> // include the LCD library
LiquidCrystal lcd(rs, en, d4, d5, d6, d7); //Initialize the LCD
```

Inside the arrangement() work, we would simply give an introduction message to be shown in the LCD screen. You can find out about interfacing LCD with STM32.

```
lcd.begin(16, 2); //We are using a 16*2 LCD
lcd.clear(); //Clear the screen
lcd.setCursor(0, 0); //At first row first column
lcd.print("Hello world"); //Print this
lcd.setCursor(0, 1); //At secound row first column
lcd.print("STM32F103C8"); //Print this
delay(2000); //wait for two secounds
lcd.clear(); //Clear the screen
```

```
lcd.setCursor(0, 0);  //At first row first column
lcd.print("USING ADC IN");//Print this
lcd.setCursor(0,1); //At secound row first column
lcd.print("STM32F103C8");//Print this
delay(2000); //wait for two secounds
lcd.clear(); //Clear the screen
```

At long last, inside our interminable circle() work, we begin perusing the simple voltage provided to the PA7 pin from potentiometer. As we talked about as of now, the microcontroller is an advanced gadget and it can't peruse voltages level straightforwardly. Utilizing SAR method the voltage level is mapped from 0 to 4096. These qualities are known as the ADC esteems, to get this ADC esteem basically utilize the accompanying line

```
int val = analogRead(A7);    // read the ADC value from pin PA7
```

Here the capacity analogRead() is utilized to peruse the simple estimation of the pin. At last we spare this incentive in a variable called "val". The kind of this variable is number since we will just get values extending from 0 to 4096 to be put away in this factor.

The subsequent stage is ascertain the voltage esteem from the ADC esteem. To do this we have the accompanying formulae

```
Voltage = (ADC Value / ADC Resolution) * Reference Voltage
```

For our situation we definitely realize that the ADC goals of our microcontroller is 4096. The ADC esteem is additionally found in the past line and put away the variable called val. The reference voltage is equivalent to the voltage at which the microcontroller is working. At the point when the STM32 board is controlled by means of USB link then the working voltage is 3.3V. You can similarly measure the working voltage by utilizing a multimeter over the Voltage Common Collector along with ground nail to the load up. So the above equation fits into our case as demonstrated as follows

float voltage = (float(val)/4096) * 3.3; //formulae to convert the ADC value to voltage

You may be mistaken for the line coast (val). This is utilized to convert the variable "val" from int information type to "drift" information type. This change is required in light in case just in case we get the consequence of val/4096 in coast we can increase it 3.3. In case the worth is gotten in number it will constantly be 0 and the outcome will likewise be zero. When we have determined the ADC worth and voltage, all that is left is to show the outcome on the Liquid Crystal Display screen which should be possible by utilizing the accompanying lines

```
lcd.setCursor(0, 0);   // set the cursor to column 0, line 0
lcd.print("ADC Val:");
lcd.print(val); //Display ADC value
lcd.setCursor(0, 1);   // set the cursor to column 0, line 1
lcd.print("Voltage:");
lcd.print(voltage); //Display voltage
```

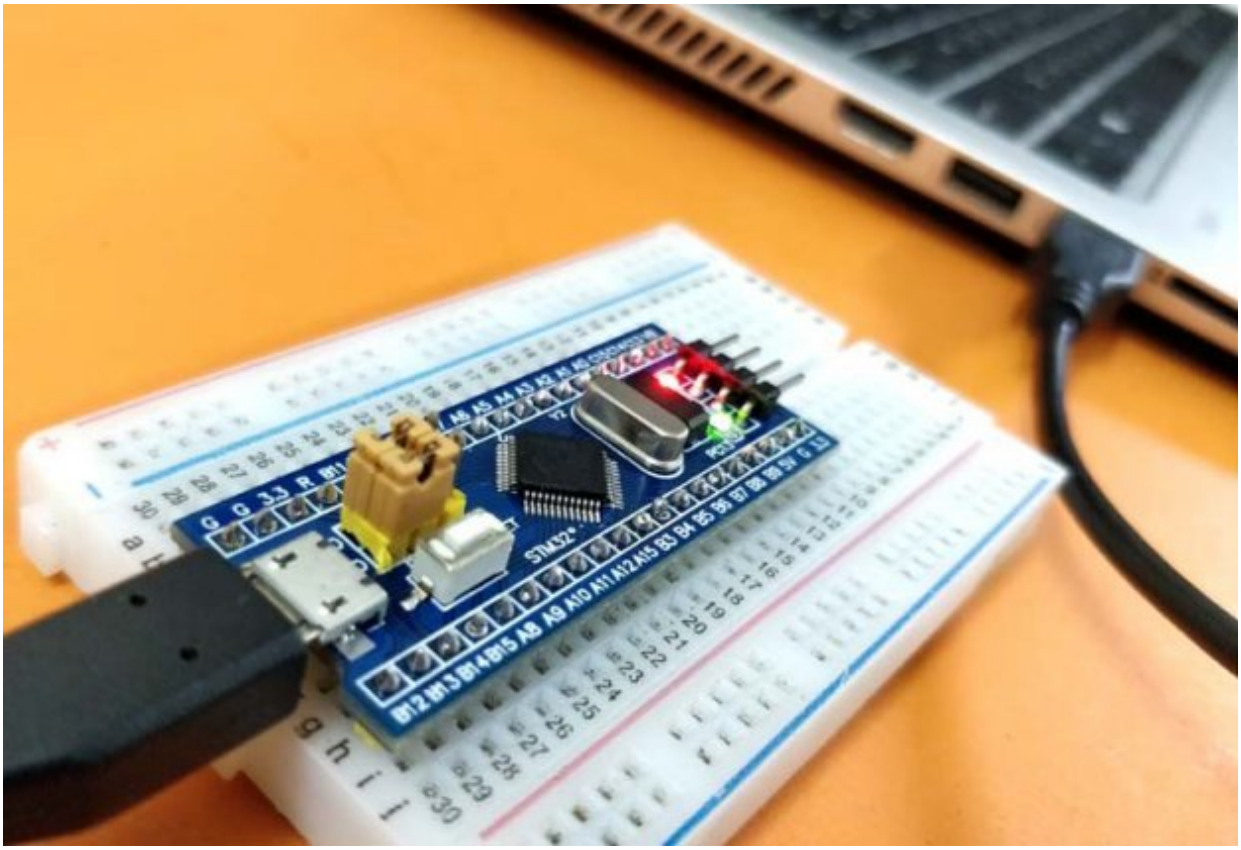
Complete code is given beneath.

Code

```
#include <LiquidCrystal.h> // include the LCD library
const int rs = PB11, en = PB10, d4 = PB0, d5 = PB1, d6 = PC13, d7 = PC14;
//mention the pin names to with LCD is connected to
LiquidCrystal lcd(rs, en, d4, d5, d6, d7); //Initialize the LCD
const int analogip = PA7;//Initialize the analog input pin
void setup()
{
 lcd.begin(16, 2); //We are using a 16*2 LCD
 lcd.clear(); //Clear the screen
 lcd.setCursor(0, 0); //At first row first column
 lcd.print("Hello world"); //Print this
 lcd.setCursor(0, 1); //At secound row first column
 lcd.print("STM32F103C8"); //Print this
```

```
  delay(2000); //wait for two secounds
  lcd.clear(); //Clear the screen
  lcd.setCursor(0, 0);  //At first row first column
  lcd.print("USING ADC IN");//Print this
  lcd.setCursor(0,1); //At secound row first column
  lcd.print("STM32F103C8");//Print this
  delay(2000); //wait for two secounds
  lcd.clear(); //Clear the screen
}
void loop()
{
  int val = analogRead(PA7);    // read the ADC value from pin A7
  float voltage = (float(val)/4096) * 3.3; //formulae to convert the ADC value
to voltage
  lcd.setCursor(0, 0);   // set the cursor to column 0, line 0
  lcd.print("ADC Val:");
  lcd.print(val); //Display ADC value
  lcd.setCursor(0, 1);   // set the cursor to column 0, line 1
  lcd.print("Voltage:");
  lcd.print(voltage); //Display voltage
}
```

10. Programming STM32F103C8 Board utilizing USB Port



The STM32 Development Board lodging the STM32F103C8 Microcontroller is getting progressively mainstream on account of its ARM Cortex M3 engineering, it has high operational speed and increasingly fringe alternatives. Likewise since, this board can be handily customized utilizing the Arduino IDE it has gotten a best decision for some specialists and designers for speedy prototyping.

In our past instructional exercise we took in the nuts and bolts of the STM32 Development Board and furthermore customized it to flicker a LED. However, there was one immense disadvantage with it. So as to program the Board we used a FTDI software engineer module and furthermore needed to flip the boot 0 jumper between and 1 position while transferring and testing a code, which is definitely an overwhelming errand. Likewise the smaller than usual USB port on the Development board was left absolutely unused. The purpose behind doing that is, the point at which the STM32 advancement board is bought it doesn't accompany an Arduino prepared

boot loader and subsequently the board won't be found by your PC when associated through the USB.

Ideally however, there exists a test boot loader created by LeafLabs for Maple scaled down sheets. This boot loader can be flashed into the STM32 once and from that point we can straightforwardly utilize the Universal Serial Bus port of the STM32 board to transfer programs simply like other Arduino sheets. Anyway this boot loader is still in formative stage at the hour of reporting this instructional exercise and isn't fitting for basic applications. Before procedures with this instructional exercise ensure you have perused the past instructional exercise to comprehend the essentials of this board including insights regarding the determinations along with pin-outs.

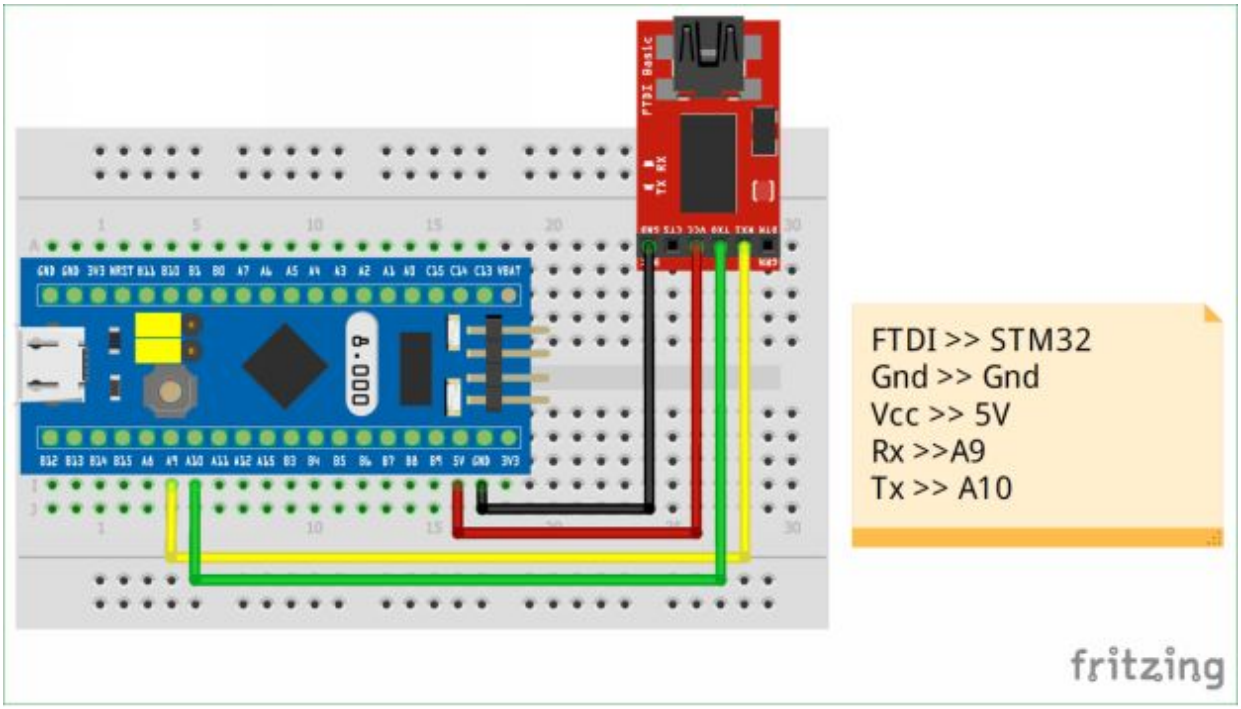
Parts Necessary

- STM32 – (BluePill) Development Board (STM32F103C8)
- FTDI Programmer
- Breadboard
- Interfacing wires
- PC with Internet

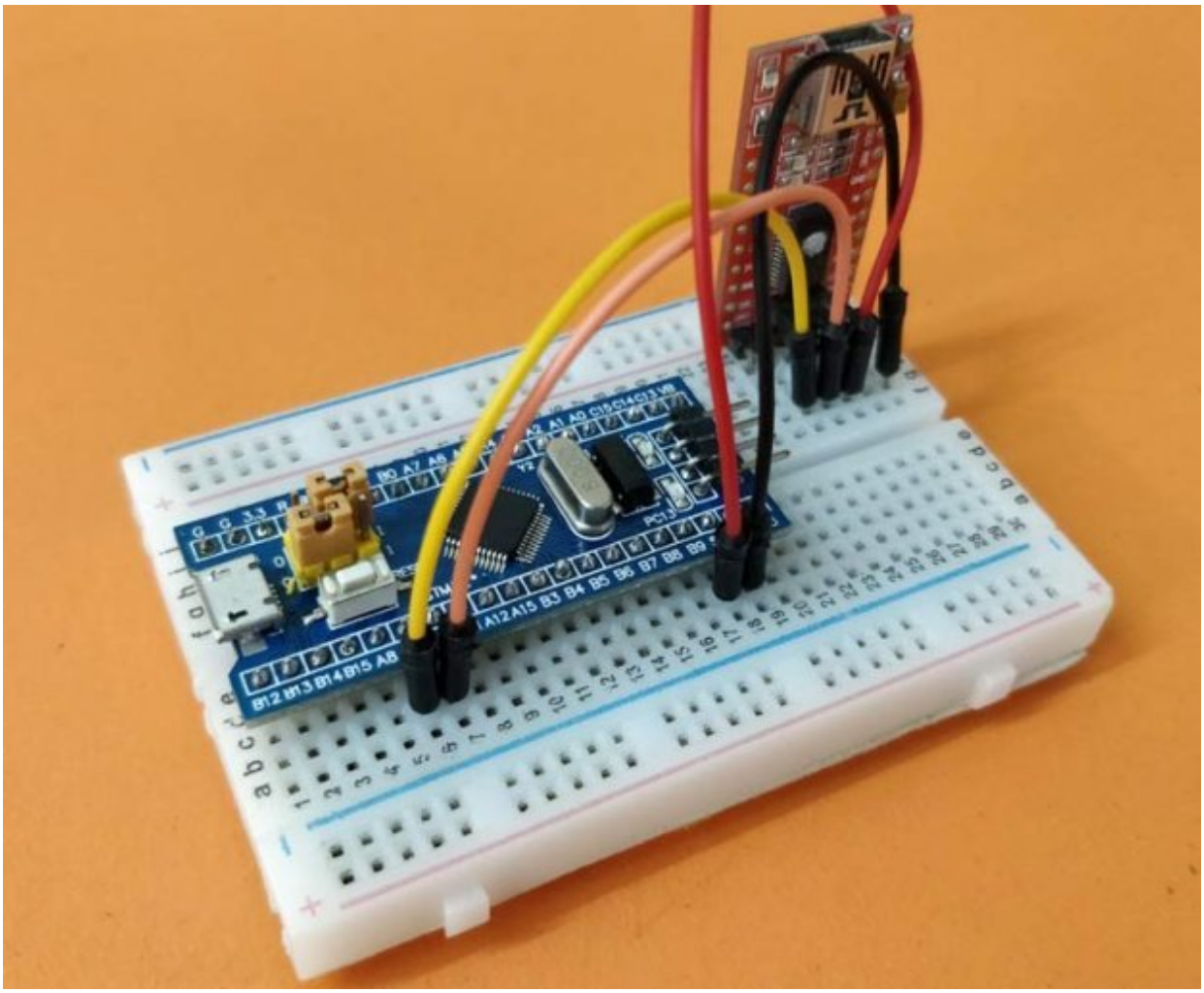
Schematic Diagram

To program the STM32 Blue Pill board legitimately through USB port we have to initially streak the Maple boot loader into the MCU. To do this

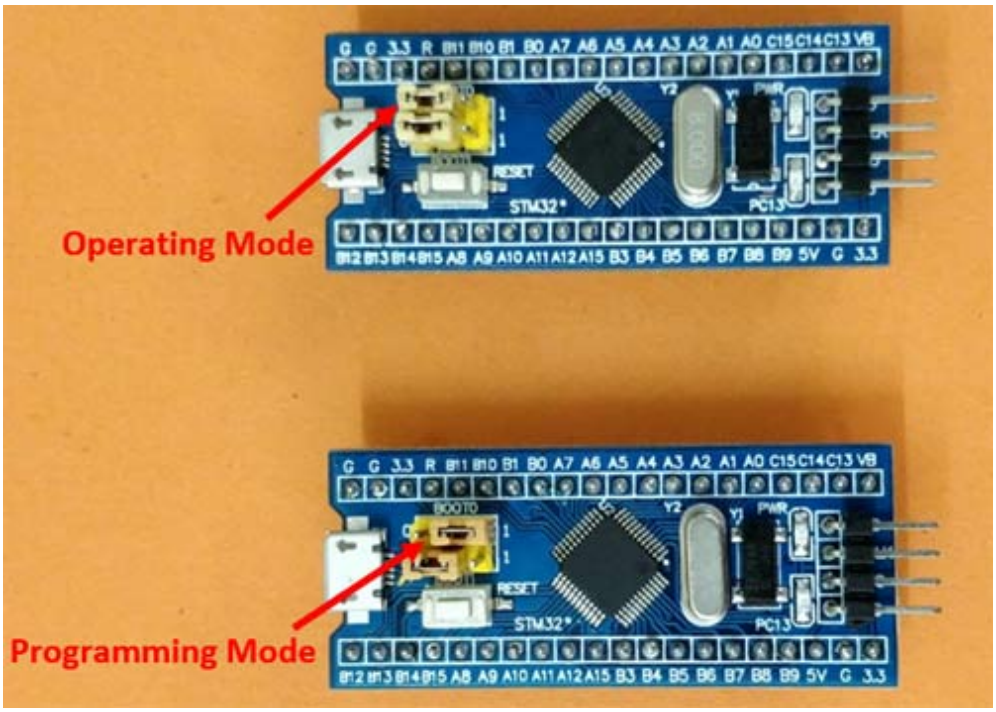
we have to utilize a Serial FTDI board. This board is associated with the Rx along with Tx pin of the STM32 as demonstrated as follows.



The Voltage Common Collector pin of the FTDI board is associated with the STM32 5V pin of intensity the board. The ground is associated with the Ground of STM32. The Rx and Tx pin of the FTDI board is associated with the A9 and A10 pin of the STM32 individually. Where the A9 is the Tx pin of STM32 MCU and the A10 is Rx pin.



Ensure the boot 0 jumper nail to the board is set to 1 (programming mode) while transferring the boot loader. When the boot loader is flashed this pin can be changed back to starting position (working mode).



Transferring the Maple Boot loader to STM32 Development board

When we have caused the above association with interface the FTDI board to your PC and follow the means to streak the boot loader into the STM32.

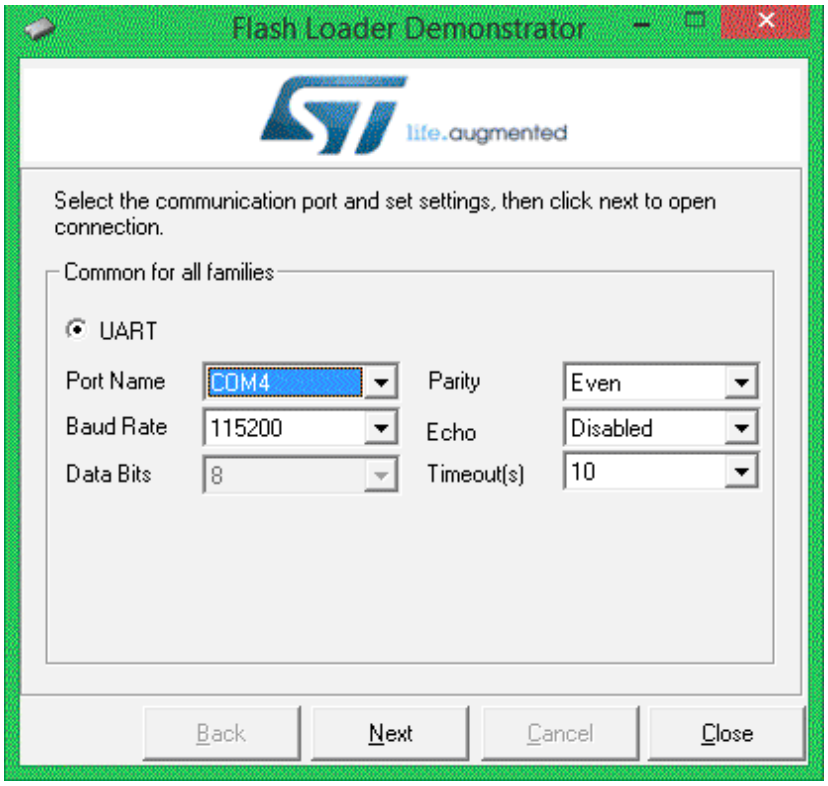
Stage 1: We need to download the boot loader program record doubles (canister document) structure the github page. There are many variants of receptacle document, for the Blue Pill board utilize this github connection and snap on the download catch to download the canister record.

Stage 2: Next we need to download and introduce the STM Flash loader programming to streak the downloaded canister document into STM32. Snap on this connect to get into the ST site and look to the base and snap on get programming

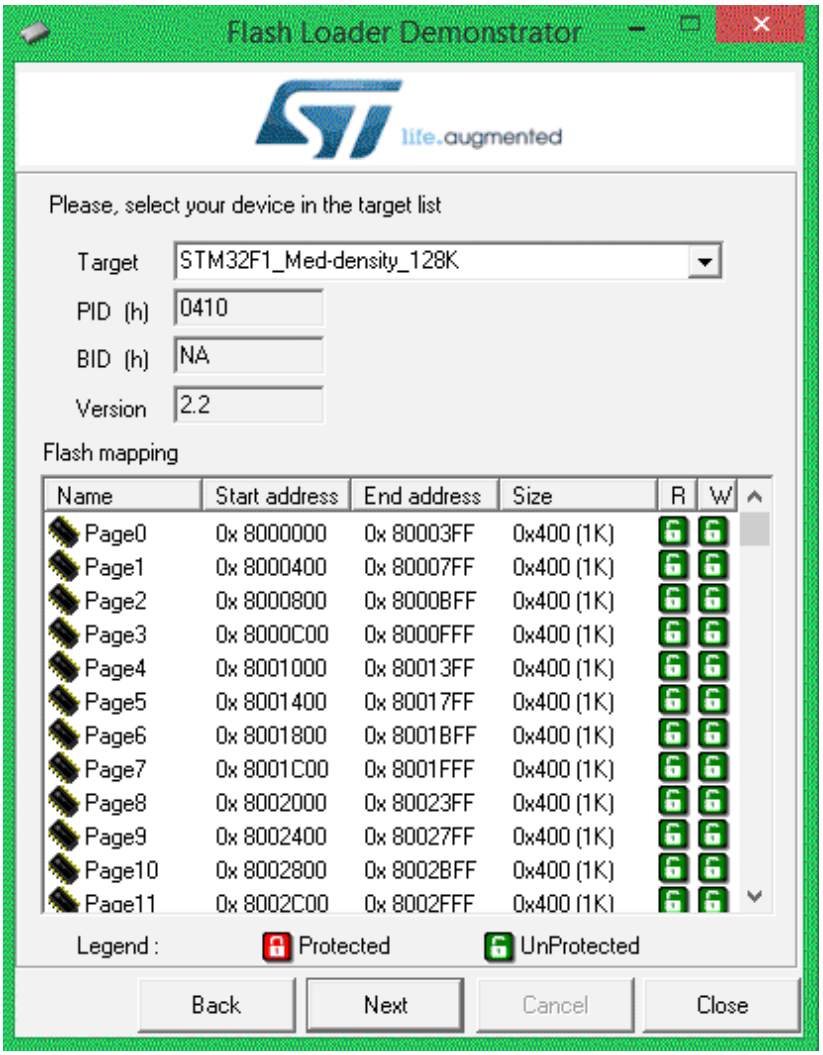
Part Number	Software Version	Marketing Status	Supplier	Order from ST
FLASHER-STM32	2.8.0	Active	ST	Get Software

Stage 3: To download the product you require to enter your E-mail address and the download connection will be sent to your E-mail. At that point follow the connection back to the site and snap on get programming again and you download will start. Indeed it's somewhat baffling however this is the manner by which it must be finished. Remember to check your spam envelope for the E-mail, at times it takes a few minutes for the E-mail to show up.

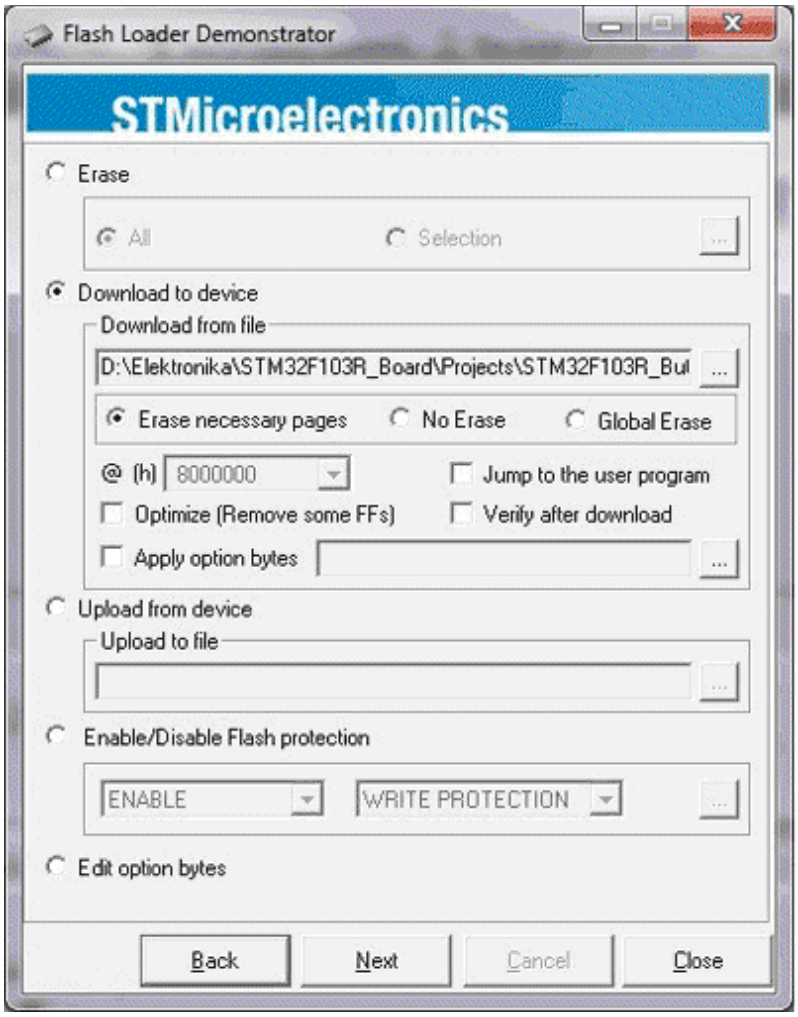
Stage 4: Once downloaded introduce the product, ensure your STM32 board is associate with your PC however FTDI board and afterward dispatch the product. The product will consequently distinguish the COM port if not utilize the Device administrator and ensure you select the right COM port number. For my situation it is COM4. Leave the remainder of the setting considering all as demonstrated as follows.



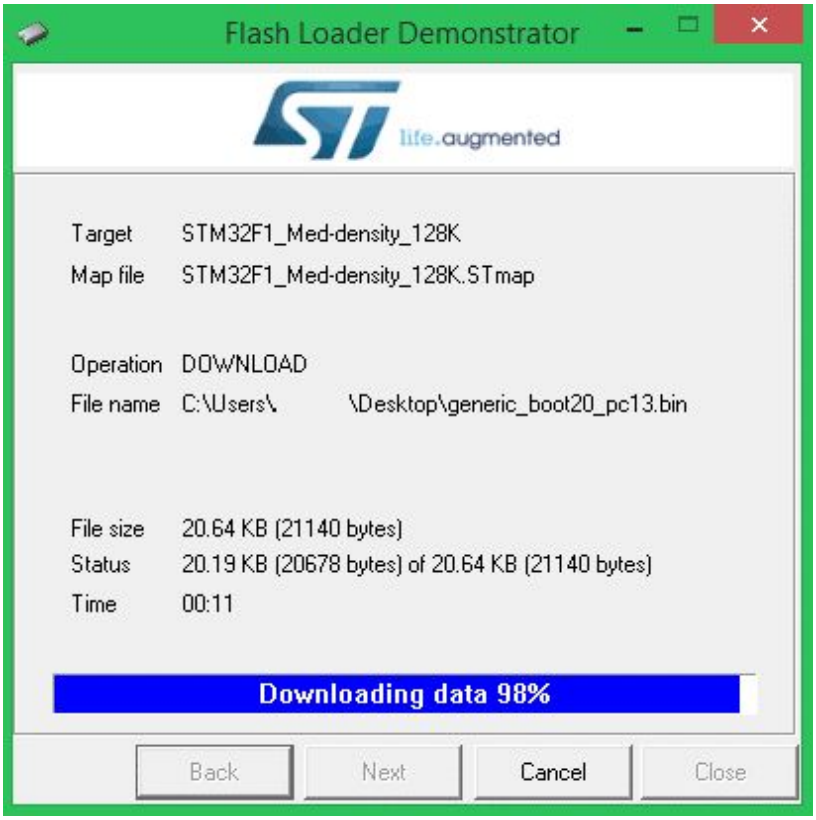
Stage 5: Click on the Next catch twice and the product will again consequently identify the board subtleties and show as demonstrated as follows. The board we are utilizing is STM32F1 with 128K blaze memory.



Stage 6: In the subsequent stage, select Download to gadget and peruse to the area where we downloaded our receptacle document in sync 1 and select it. Snap on straightaway.



Stage 7: The product will download some necessary records as appeared beneath and will at that point start the way toward blazing.



Stage 8: Once the glimmering is finishes effectively, we will get the underneath screen. Snap on close and leave the application. We have flashed the STM32 board with Arduino boot loader effectively. Presently we need to set up the Arduino IDE and introduce the drivers before we can program the STM32 board.

Setting up the Arduino IDE and Installing the Drivers

Follow the underneath steps to download and set up the Arduino IDE to be utilized with the STM 32 Development board.

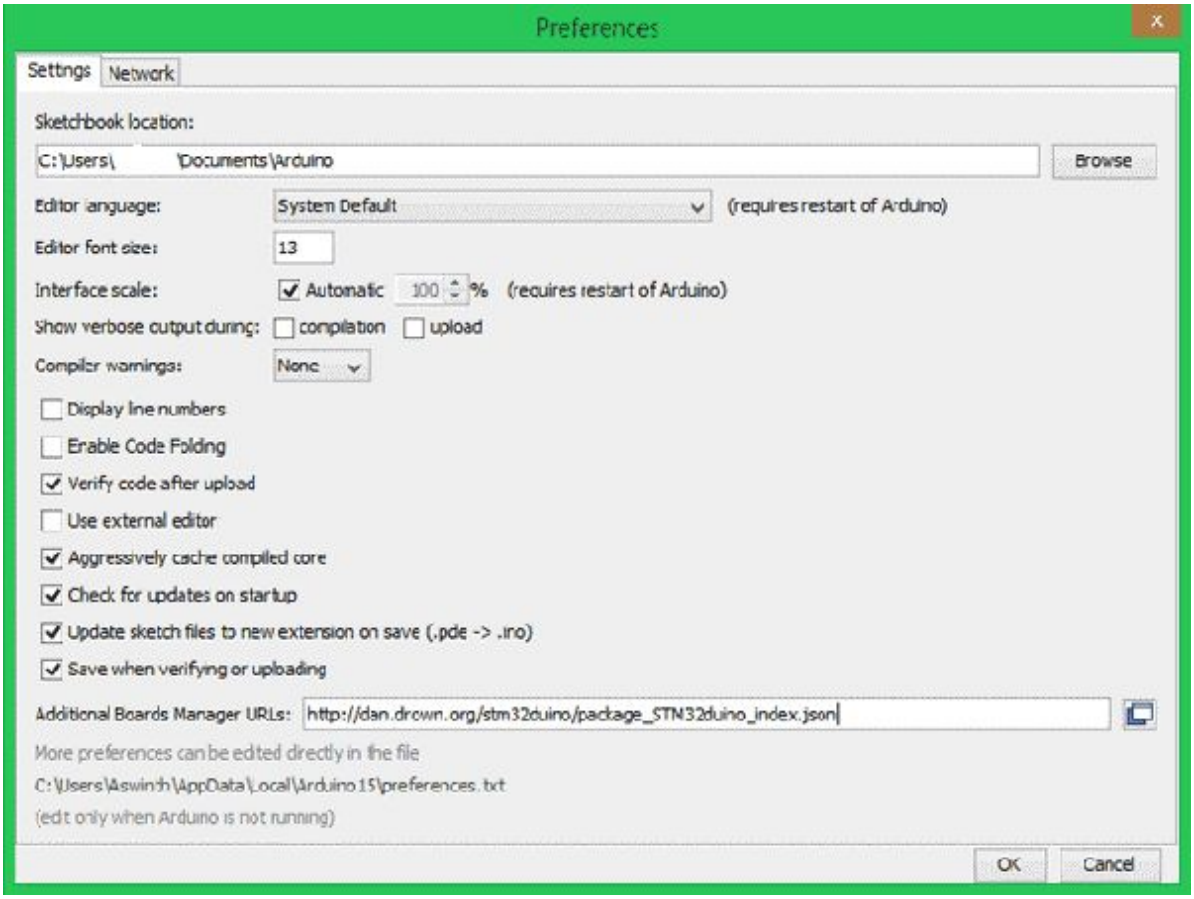
Stage 1:- If you have not yet introduced the Arduino IDE, download and introduce it from this connection. Ensure you select your right working framework.

Stage 2:- After Installing the Arduino IDE open and download the necessary bundles for the STM32 board. This should be possible by choosing File - > Preferences.

Stage 3:- Clicking on Preferences will open the underneath demonstrated discourse box. In the extra Boards Manager URL content box glue the underneath interface

http://dan.drown.org/stm32duino/package_STM32duino_index.json

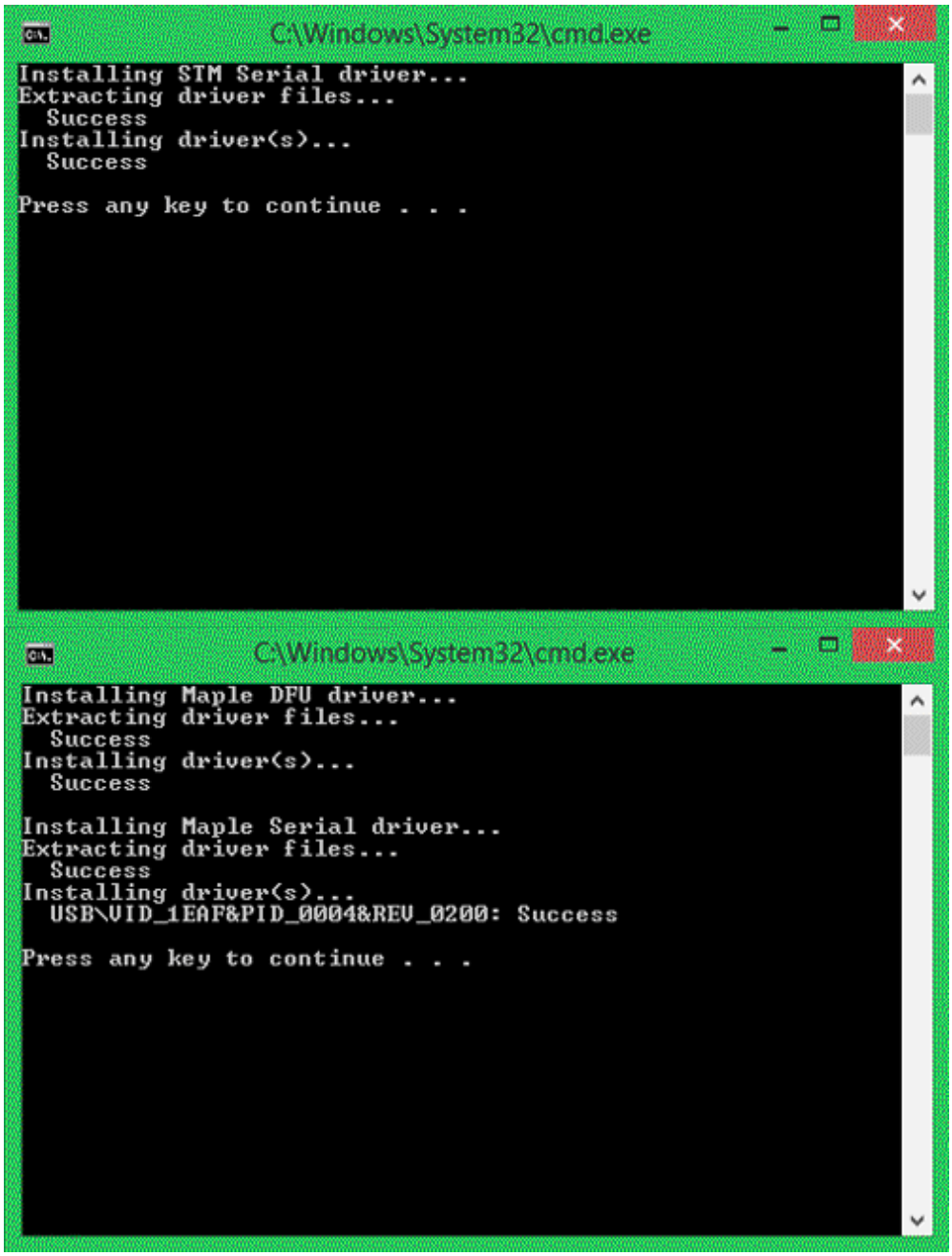
what's more, press OK.



Stage 4:- Now go to Tool - > Boards - > Board Manager. This will open the Boards director discourse box, scan for "STM32F1" and introduce the bundle that shows up.

Stage 5: After the bundle is introduced, explore to C:\Program Files (x86)\Arduino\hardware\Arduino_STM32-master\drivers\win where you will discover install_drivers.bat and install_STM_COM_drivers.bat.

Stage 6: Click on both the bat records and introduce the drivers. You will get a DOS screen as demonstrated as follows.

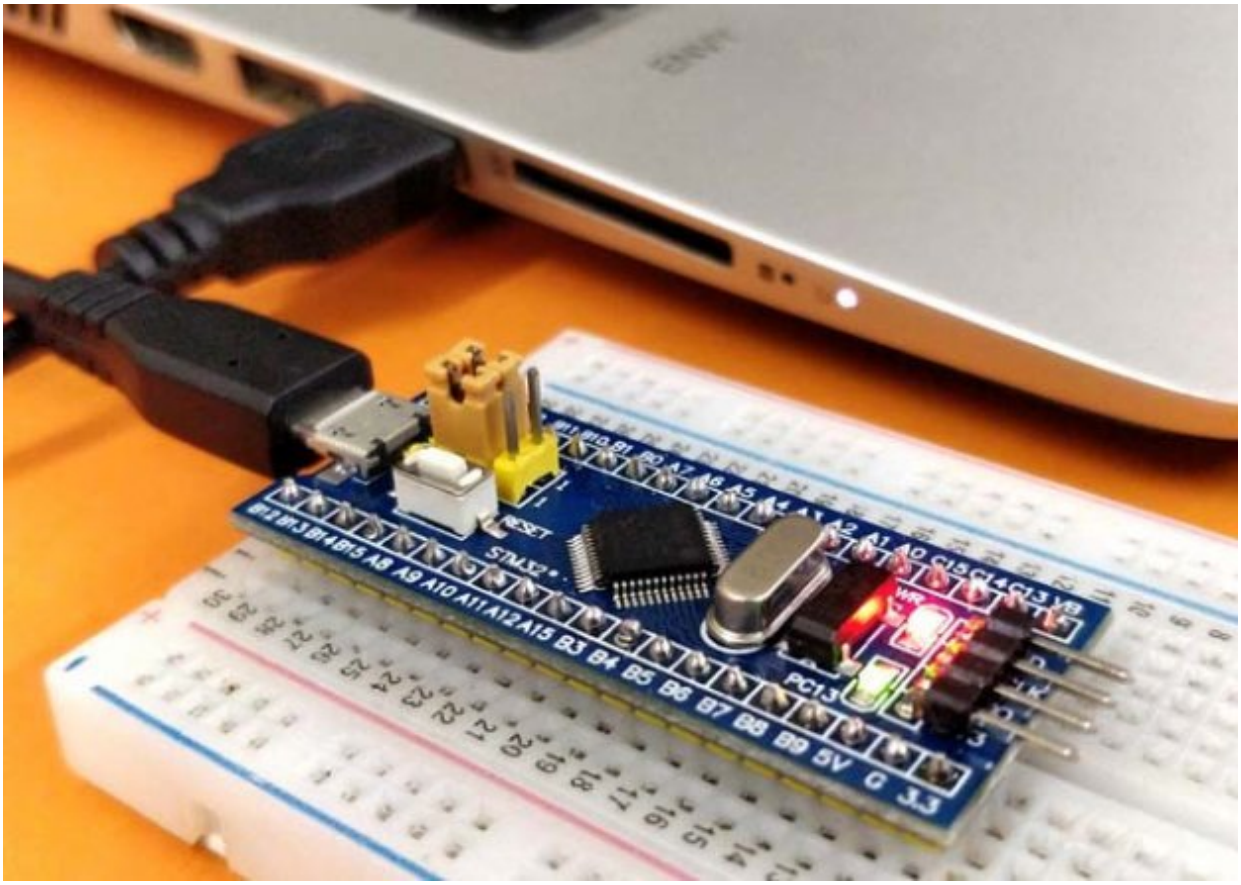


Presently the Arduino IDE is set up for programming STM32 (Blue Pill) Development Board and the drivers are likewise introduced.

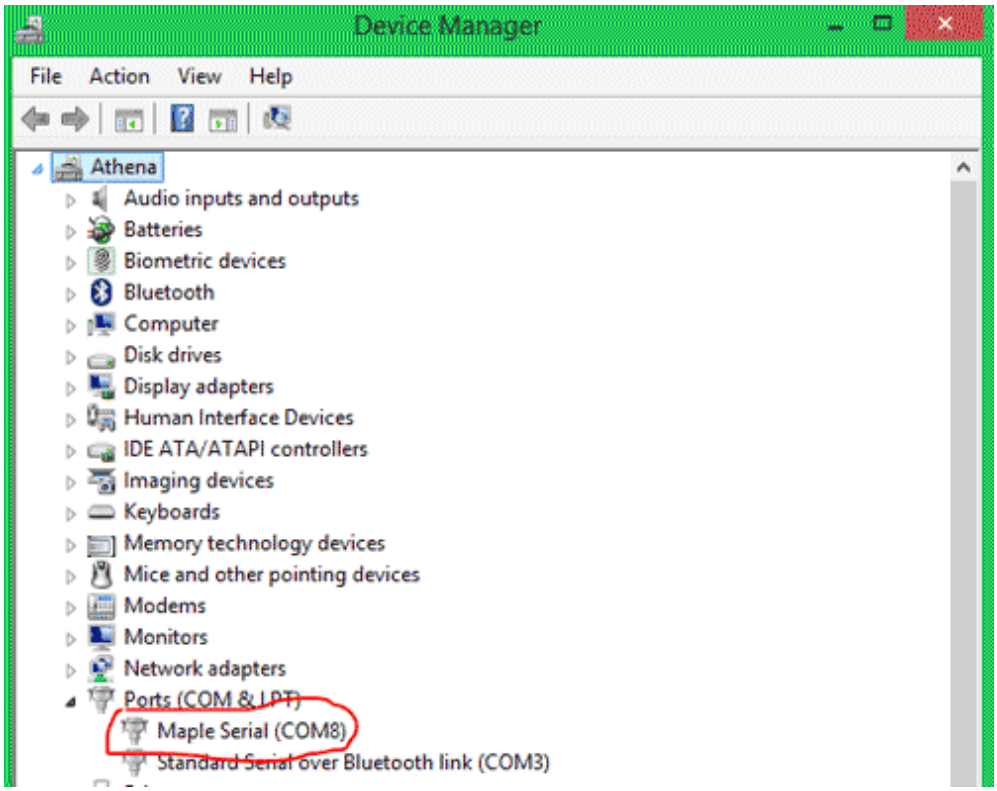
Programming STM32 (Blue Pill) Directly Through USB Port

Evacuate the FTDI board and all the current associations from you STM32. Simply utilize the miniaturized scale Universal Serial Bus port on

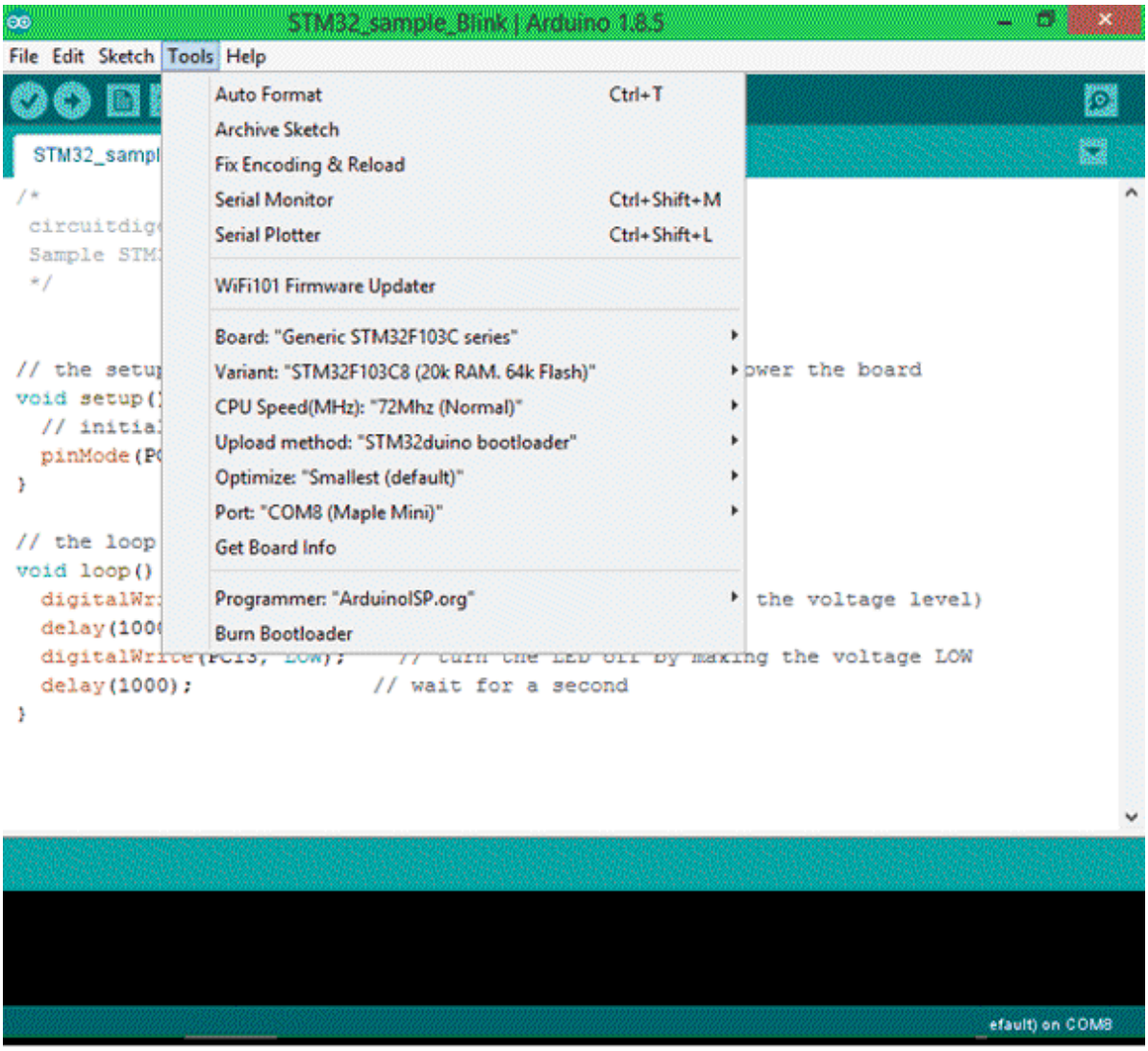
the STM32 board to associate it to the PC as demonstrated as follows. Ensure the jumper 0 pin is situated back at 0 (Operating mode). In the future we need not flip the jumper any longer to transfer and run the projects.



You PC ought to have the option to find the Board now. Sit tight for some time in case you see any extra drivers getting introduced. At that point get into Device director and check if your STM32 board is found under the COM and port segment as demonstrated as follows. Mine is associated with COM8 with the name Maple Mini.



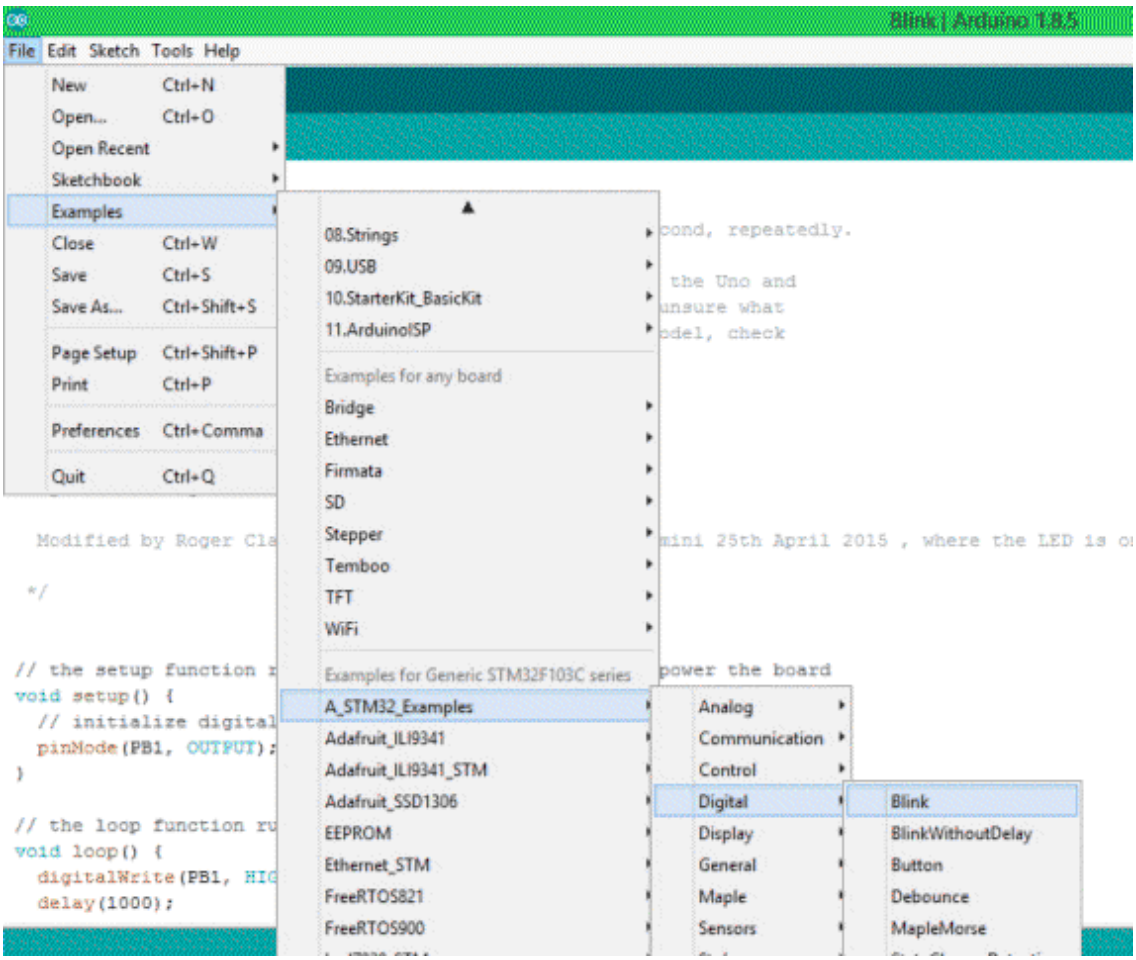
Go to Tools and look down to locate the Generic STM32F103C arrangement as demonstrated as follows. At that point ensure the variation is 64k Flash sort, CPU speed is 72MHz along with modify the transfer technique to STM32duino Bootloader. Likewise select the right COM port as per the one on your gadget supervisor.



After all the progressions are made, check the base right corner of the Arduino IDE along with you should view the accompanying setting being set. My STM32 board is associated with COM8 yet yours strength vary

Presently the Arduino IDE is prepared to program the STM 32 Blue Pill Development Boards. Let us transfer the Sample Blink Program from the

Arduino IDE to the STM32 Blue Pill board to ensure everything is working appropriately. The model program can be found at



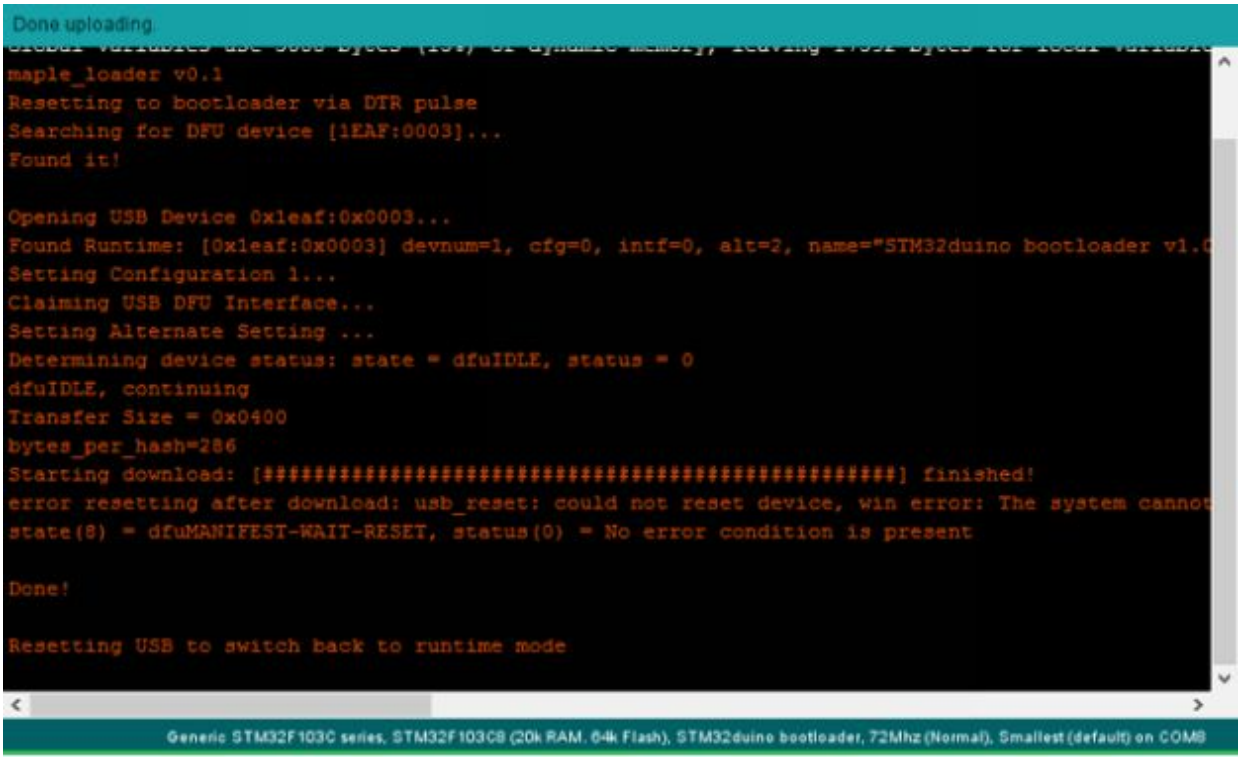
In the model program that opens, we need to roll out a little improvement. As a matter of course the program will be composed for PB1 yet on our board the on-board LED is associated with PC13 so supplant all PB1 with PC13 and we are a great idea to continue. The total model program which is adjusted can likewise be found at the base of this page.

The code inside the circle work alone is demonstrated as follows, where we can see that the PC13 pin is kept HIGH (on) for 1000 millisecond and afterward killed LOW (off) for another 1000 millisecond and this is accomplished for boundless occasions since it is in circle work. In this way

the LED has all the earmarks of being flickering with an interim of 1000 millisecond.

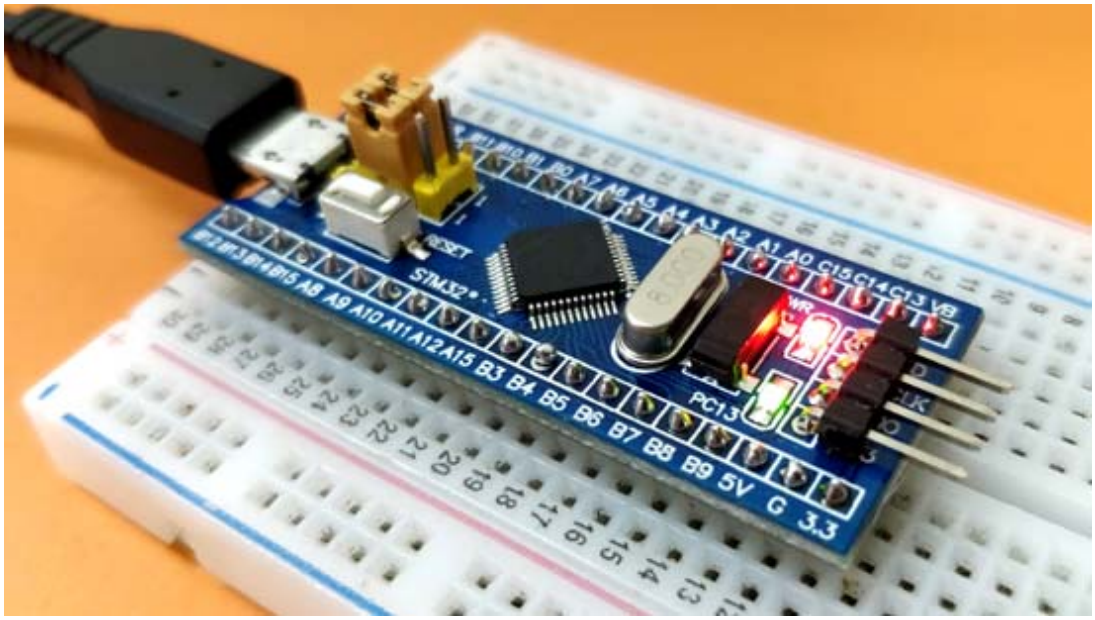
```
    digitalWrite(PC13, HIGH);     // turn the LED on (HIGH is the voltage level)
    delay(1000);                  // wait for a second
    digitalWrite(PC13, LOW);      // turn the LED off by making the voltage low
    delay(1000);                  // wait for a second
```

Press the transfer button on the Arduino IDE along with your program ought to get incorporated and transferred. In the event that everything has filled in true to form, at that point you should see the accompanying on your Arduino IDE reassure.



On the off chance that the Program has been transferred effectively, at that point you should see the Green LED flickering at a 1 second interim. You can similarly mess with the program to increment or diminishing the

postponement. Presently you can begin utilizing the STM32 (Blue Pill) Development board like some other Arduino sheets, that is you no longer need not change the situation of jumpers or utilize outside equipment to transfer and test programs.



Expectation you comprehended the instructional exercise and thought that it was valuable to begin with STM32 Board. In case you have any issue leave them in the remark area, likewise mention to me what ventures we should attempt with this STM32 load up in future.

thank you