Beginning Level of Circuit board trainer

Flickering LED, STM32CubeMX and TrueSTUDIO - Simple LED Control , Humidity Sensor,Sequential Communication, Send/Receive SMS with STM32F103C8 and SIM800C

BEGINNING LEVEL OF CIRCUIT BOARD TRAINER PROJECTS

Flickering LED, STM32CubeMX and TrueSTUDIO - Simple LED Control , Humidity Sensor,Sequential Communication, Send/Receive SMS with STM32F103C8 and SIM800C etc...,

Anbazhagan K

CONTENTS

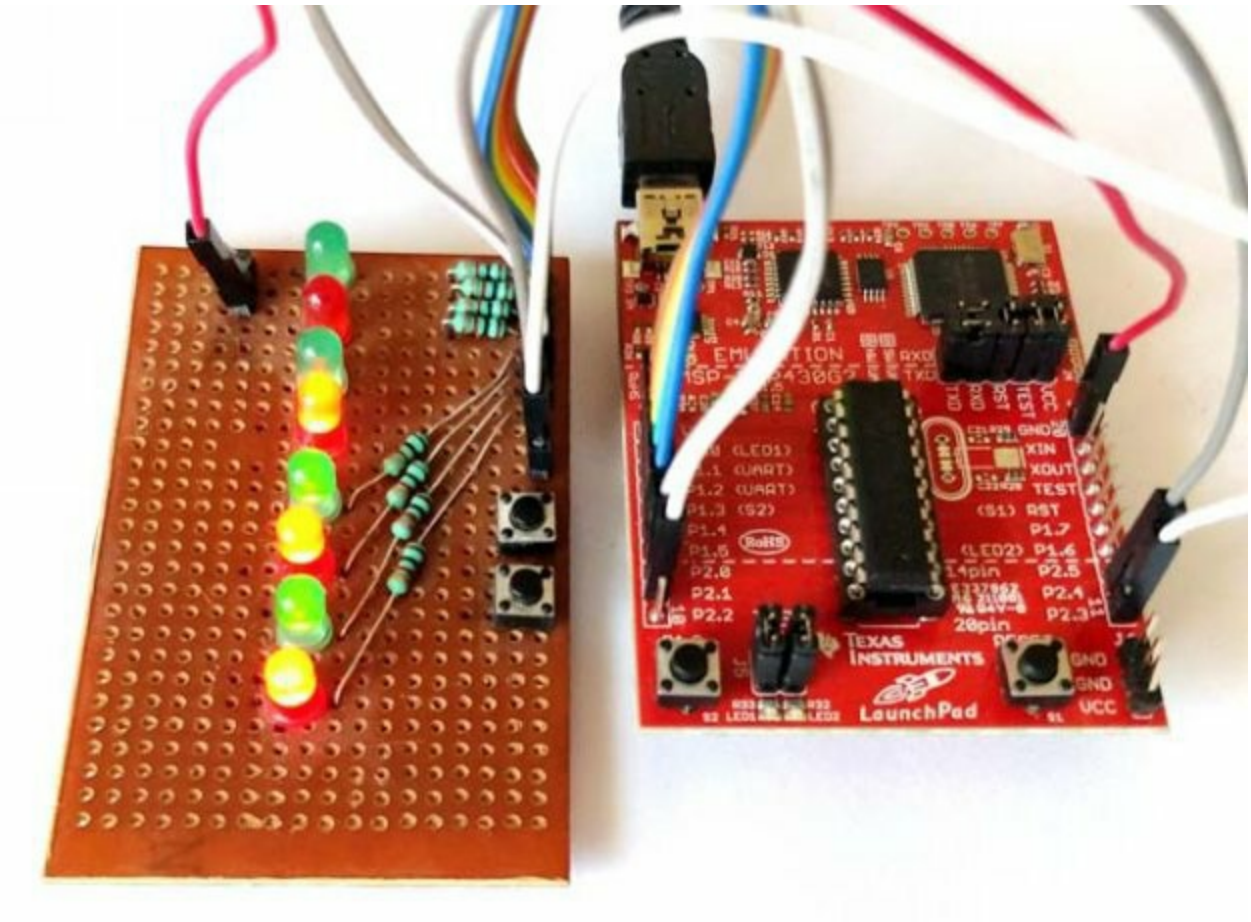
ACKNOWLEDGMENTS

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. FLICKERING LED SEQUENCE WITH MSP430G2: USING DIGITAL READ/WRITE PINS



This is the second instructional exercise of a succession of instructional exercise wherein we are taking in the MSP430G2 LaunchPad from Texas Instruments utilizing the Energia IDE. In the last Blinky LED instructional exercise we acquainted our self with the LaunchPad Development Board and the Energia IDE, we likewise transferred our first program which is to flicker the on board LED at a normal interim.

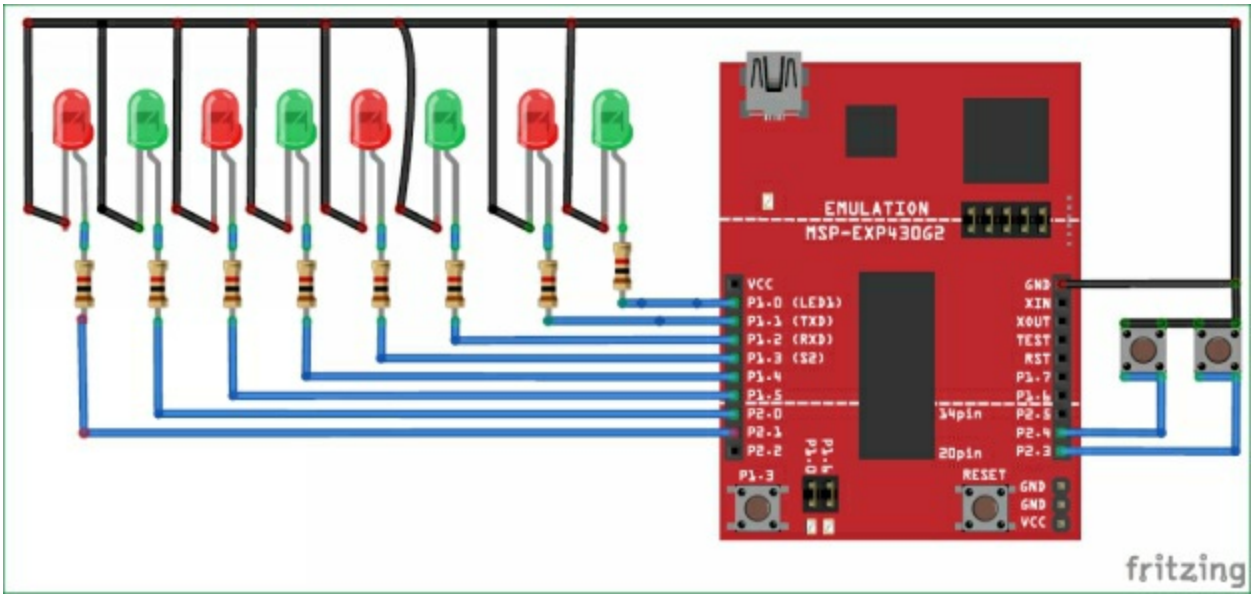
Here we will figure out how to use the Digital Read and Digital Write choice to peruse the status of an info gadget like a switch, and control various yields like LED's. Toward the finish of this instructional exercise you would have figured out how to function with Digital Inputs and yields, which can be used to interface numerous advanced sensors like IR sensor, PIR sensor and so forth and furthermore to turn on or off yields like LED, Buzzer and so on. Sounds intriguing right!!? We should begin.

Material Required:

- MSP430G2 LaunchPad
- Driven of any shading - 8
- Switch - 2
- 1k Resistor - 8
- Associating wires

Circuit Diagram:

In our past instructional exercise, we view that the platform itself accompanies two LED and a switch on the board. In any case, in this instructional exercise we are going to require more than that, as we are wanting to sparkle eight LED lights in an arrangement when a catch is squeezed. We will likewise change the grouping when another catch is squeezed just to do it intriguing. So we need to manufacture a circuit with 8 LED lights and two switches, the total circuit chart can be found underneath.



Here the 8 LED's are the yields and the two switches are the sources of info. We can interface these to any I/O nail to the board however I have associated the LRD's from pin P1.0 to P2.1 and change 1 and 2 to stick P2.4 and P2.3 separately as appeared previously.

All the cathode pins of the LED are attached to ground along with the anode pin is combined with the I/O sticks through a resistor. This resistor is known as a Current constraining resistor, this resistor isn't compulsory for a MSP430 in light of the fact that the most extreme current it's I/O pin can source is just 6mA and the voltage on pin is 3.6V as it were. Anyway it is a decent practice to utilize them. At the point when any of these advanced pins go high the individual LED will turn on. In case you can review the last instructional exercises LED program, in this point you will recollect that digitalWrite (Light Emitting Diode_pin_name, HIGH) will make the Light Emitting Diode sparkle along with digitalWrite (LED_pin_name, LOW) will turn of the LED.

The switches are the information gadget, one finish of the switch is associated with the ground terminal along with the other is associated with computerized pins P2.3 and P2.4. This implies at whatever point we press the switch the I/O pin (2.3 or 2.4) will be grounded and will be without left if the catch isn't squeezed. Let us perceive how we can utilize this course of action while programming.

Programming Explanation:

The program must be composed to control the 8 LED in a grouping way when the switch 1 is squeezed and afterward when switch 2 is squeezed the arrangement must be changed. Further underneath I will clarify the program line by line with the goal that you can without much of a stretch get it.

As consistently we should begin with the void arrangement () work inside which we would pronounce the pins that we are utilizing is info or yield pin. In our program the 8 LED pins are yield and the 2 switches are inputs. These 8 LEDs are associated from P1.0 to P2.1 which is nail number 2 to 9 to the

board. At that point the switches are associated with pin P2.3 and Pin 2.4 which is pin number 11 and 12 individually. So we have pronounced the accompanying in void arrangement ()

```
void setup() {
  for (int i = 2; i <= 9; i++) {
    pinMode(i, OUTPUT);
  }
  for (int i = 2; i <= 9; i++) {
    digitalWrite(i, LOW);
  }
  pinMode (11, INPUT_PULLUP);
  pinMode (12, INPUT_PULLUP);
}
```

As we probably am aware the pinMode() work announces the pin to be yield or input and the digitalWrite() work makes it high (ON) or low (OFF). We have utilized a for circle to make this announcement to lessen the number for lines. The variable "I" will be increased from 2 to 9 in the for circle and for every augmentation the capacity inside will be executed. Something else that may befuddle you is the expression "INPUT_PULLUP". A pin can be announced as contribution by simply calling the capacity pinMode (Pin_name, INPUT) however here we have utilized an INPUT_PULLUP rather than an INPUT and the 2 of them have a recognizable change.

At the point when we are utilizing any microcontroller pins, the pin ought to either be associated with low or to high. For this situation the pin 11 and 12 is combined with the switch which will be associated with ground when squeezed. Be that as it may, when the switch isn't squeezed the pin isn't associated with anything this condition is known as a coasting pin and it is awful for microcontrollers. So to keep away from this we either go through a draw or pull-down resistor to hold the pin to a state when it gets in gliding. In MSP430G2553 Microcontroller the I/O pins have a draw up resistor in-manufactured. To utilize that we should simply call INPUT_PULLUP rather than INPUT during statement simply like w have done previously.

Presently gives step access to the void circle() work. Whatever is written in this capacity will be executed for ever. The initial phase in our program is to check if the switch is squeezed and whenever squeezed we should begin flicker the LEDs in arrangement. To check if the catch is squeezed the accompanying line is utilized

```
if (digitalRead(12) == LOW)
```

Here the new capacity is the digitalRead() work, this capacity will peruse the status of a computerized pin along with will return HIGH(1) when the pin is getting some voltage along with will return low LOW(0) when the pin is grounded. In our equipment, the pin will be grounded just when we press the catch else it will be high since we have utilized a draw up resistor. So we utilize the if articulation to check if the catch was squeezed.

When the catch is squeezed we get into the limitless while (1) circle. This is the place we begin squinting the LEDs in arrangement. A limitless while circle is appeared beneath and whatever is composed inside the circle will run always until a break; articulation is utilized.

```
whiel(1){

}
```

Inside the interminable while we check for the status of the second switch which is combined with pin 11.

In the event that this switch is squeezed we squint the LED in one specific succession else we will flicker it in another arrangement.

```
if (digitalRead(11) == LOW)
      {
       for (int i = 2; i <= 9; i++)
        {
         digitalWrite(i, HIGH);
         delay(100);
        }
          for (int i = 2; i <= 9; i++)
              digitalWrite(i, LOW);
      }
```

To squint the LED in grouping we again utilize the for circle, yet this time we utilize a little deferral of 100 milliseconds utilizing the delay(100) work with the goal that we can see the LED getting high. To make just each LED sparkle in turn we additionally utilize another for circle to kill all the LED. So we turn on a drove hang tight for quite a while and afterward turn off all the LED then augmentation the tally turn on the LED sit tight for quite a while and the cycle proceeds. Be that as it may, this will occur as long as the

subsequent switch isn't squeezed.

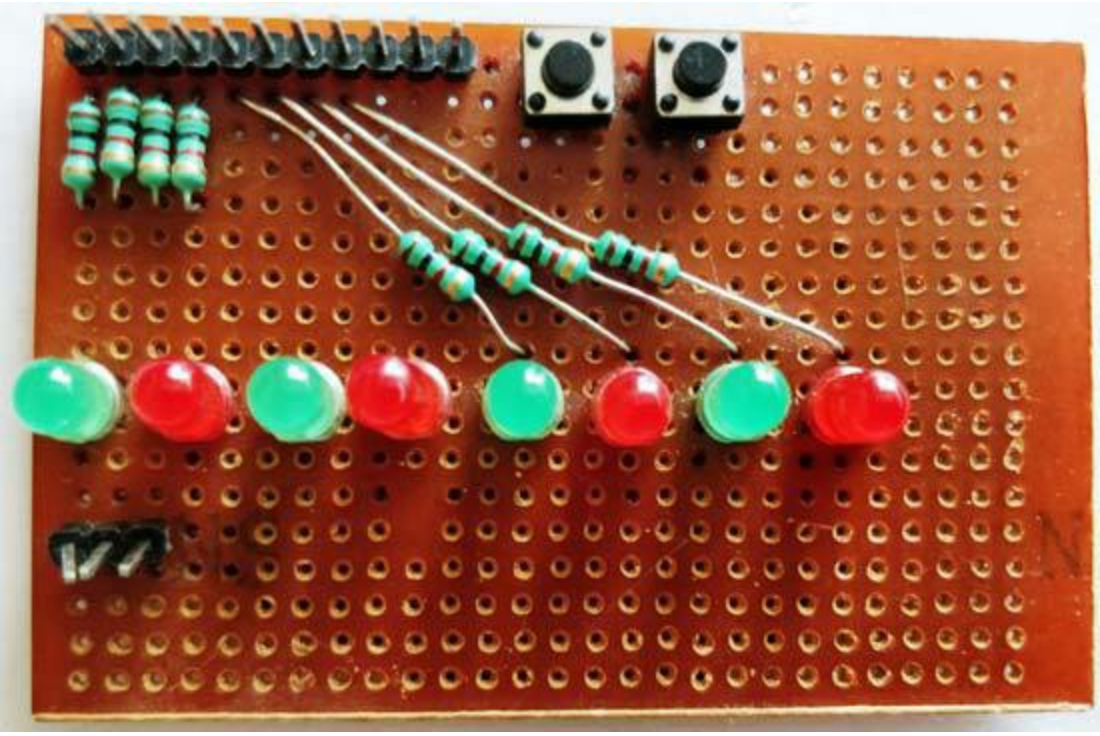
In the event that the subsequent switch is squeezed, at that point we modify the arrangement, the program will be pretty much the equivalent expect for the grouping of which the LED is turned on. The lines are appeared underneath give investigating and calculating a shot what has been changed.

```
else
  {
    for (int i = 9; i >= 2; i--)
    {
     digitalWrite(i, HIGH);
     delay(100);
    }
            for (int i = 2; i <= 9; i++)
        digitalWrite(i, LOW);
  }
```

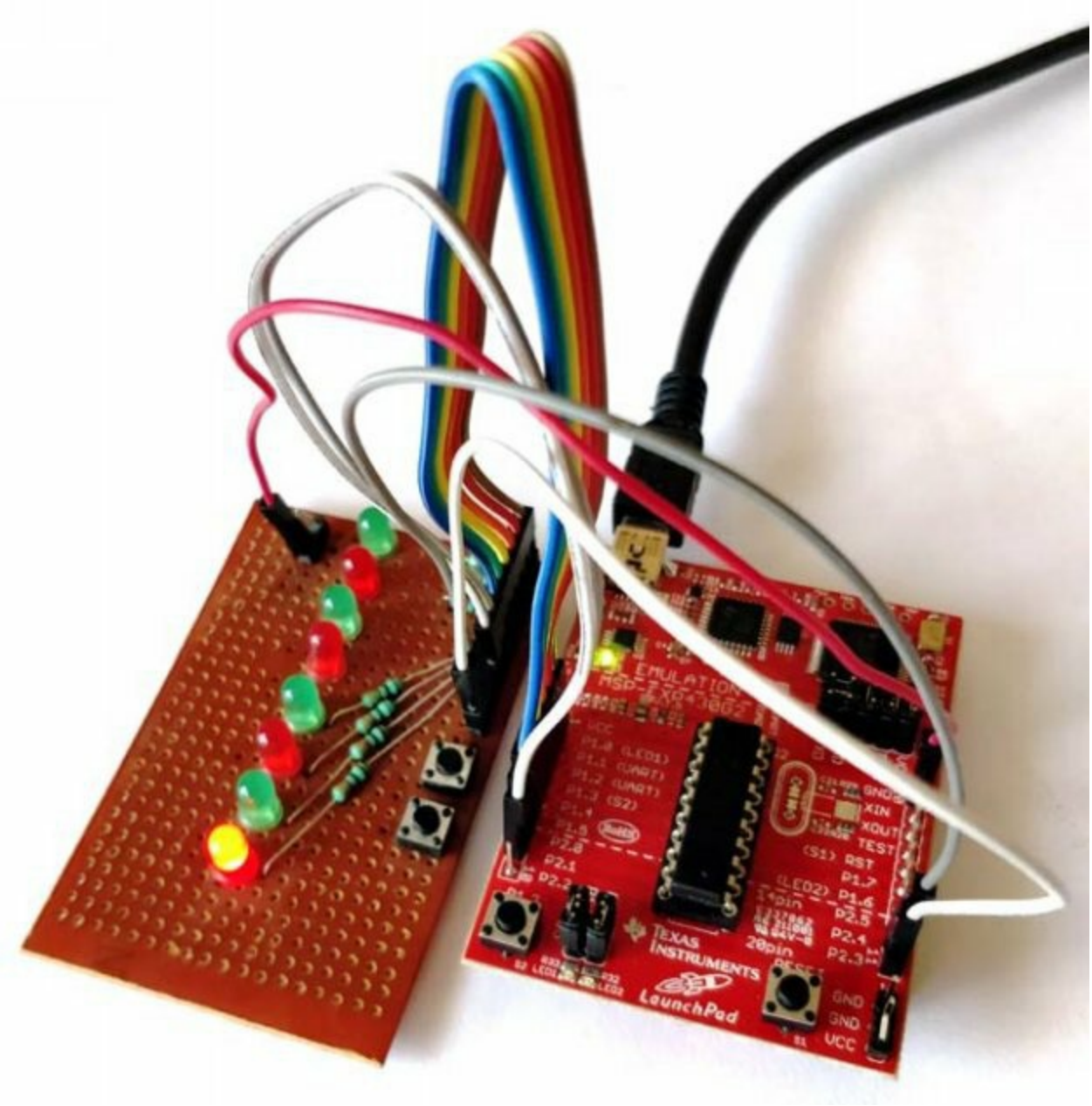
Truly, the for circle has been modified. Already we made the LED to sparkle from number 2 and as far as possible up to 9. Be that as it may, presently we are gonna to begin from number 9 and decrement right down to 2. In this way we can view if the switch is squeezed or not.

Equipment Setup for Blinking LED Sequence:

Alright enough of all the hypothesis and programming part. Let's, get a few parts and perceive how this program glances in real life. The circuit is exceptionally basic and consequently can be handily based on a breadboard. Be that as it may, I have bound the LED and switches on perf board just to do it look flawless. The perf board that I welded is demonstrated as follows.



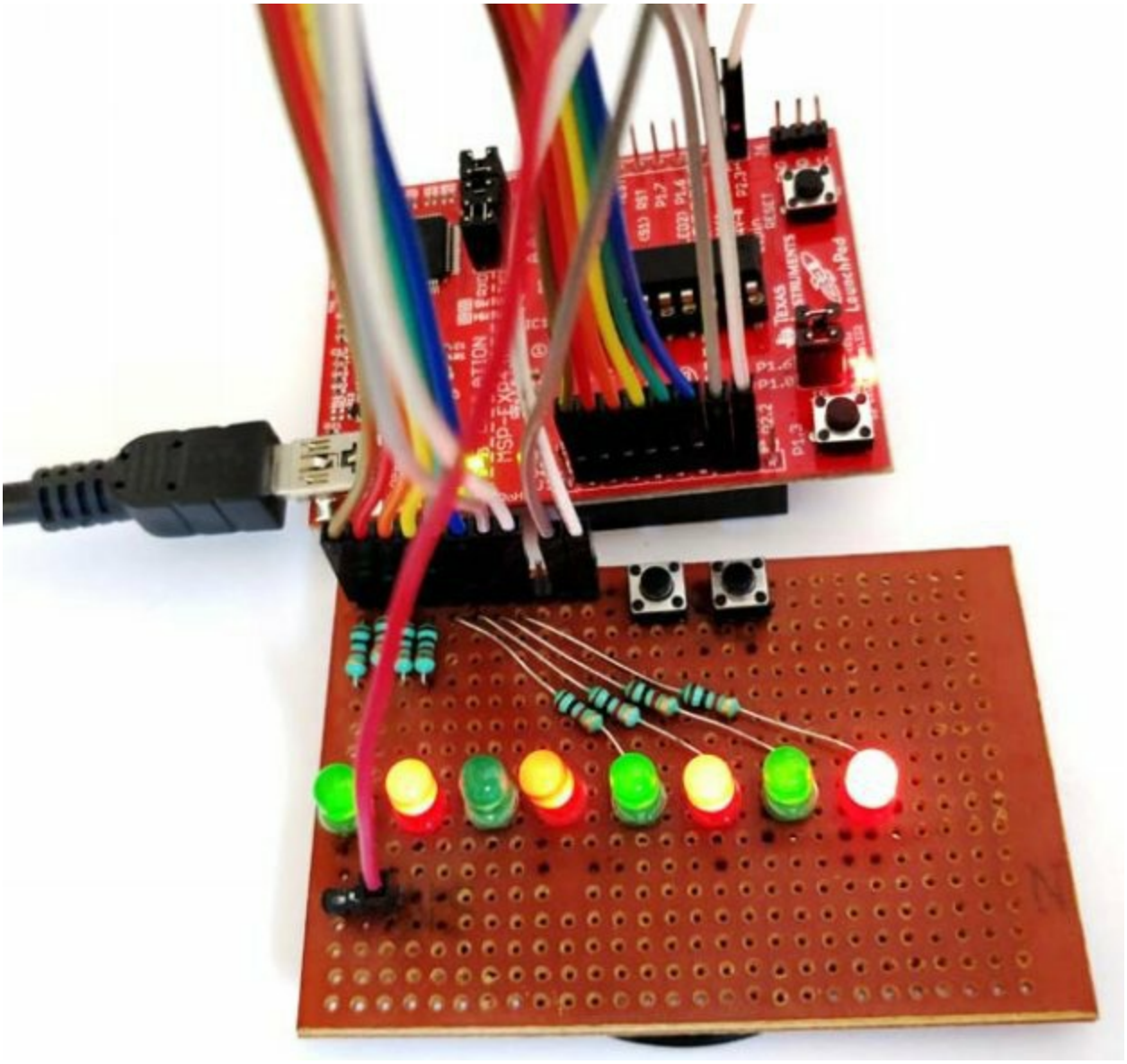
As should be obvious we have the yield pins of the LED and change taken out as connector pins. Presently we have utilize the female to female connector wires to associate the LEDs and changes to out MSP430 LaunchPad board as appeared in the image underneath.



Transferring and Working:

When you are finished with the equipment, simply associate you MSP430 board to your PC and open the Energia IDE and utilize the program given toward the finish of this page. Ensure the correct board and COM port are chosen in the Energia IDE and snap on the Upload button. The program ought to get arranged effectively and once transferred is will show "Done Uploading".

Presently press the catch 1 on the board and the LED should illuminate in grouping as demonstrated as follows



You can similarly hold the subsequent catch to check if the grouping is getting changed. In the event that you are happy with the outcomes you can take a stab at rolling out certain improvements in the code like modifying the postpone time changing the arrangement and so forth. This will assist you with learning and see better.

Expectation you have comprehended the instructional exercise and picked up something valuable with it. How about we meet in another instructional exercise where will figure out how to peruse simple voltages utilizing our MSP30 platform.

Code

```
/*
  TUTORIAL 2 - Learning to use I/O
  This program will control 8 LEDs based ont he input from two push button
  LED should be connected form P1.0 to P2.1 (pin 2 to 7)
  Switch is connected to P2.3 and P2.4 (pin 8 and 9)
*/
```

```
void setup() {
 for (int i = 2; i <= 9; i++) {
   pinMode(i, OUTPUT);
 }
 for (int i = 2; i <= 9; i++) {
   digitalWrite(i, LOW);
 }


  pinMode (11, INPUT_PULLUP);
 pinMode (12, INPUT_PULLUP);
}
```

```
// the loop routine runs over and over again forever:
void loop() {
```

```
 if (digitalRead(12) == LOW)
{
   while (1)
   {
     if (digitalRead(11) == LOW)
     {
      for (int i = 2; i <= 9; i++)
       {
        digitalWrite(i, HIGH);
        delay(100);
```

```
      }
        for (int i = 2; i <= 9; i++)
          digitalWrite(i, LOW);
    }
    else
    {
     for (int i = 9; i >= 2; i--)
     {
      digitalWrite(i, HIGH);
      delay(100);
     }
            for (int i = 2; i <= 9; i++)
          digitalWrite(i, LOW);
    }
  }
}
}
```

2. BEGINNING WITH MSP430G2 UTILIZING ENERGIA IDE – BLINKING A LED

The MSP-EXP430G2 is a Development Tool a.k.a LaunchPad gave by the Texas Instruments to learn along with rehearse on the best way to utilize their Microcontrollers. This board falls under the MSP430 Value Line class where we can program all the MSP430 arrangement Microcontrollers. This delightful sparkly Red board is interesting to learn on account of the very certainty that it has a place with TI (Texas Instruments). Figuring out how to utilize TI Microcontrollers would insubordinately be a relentless apparatus up

in our sleeve since TI is extremely immense along with has a wide verity of MCU's to browse at a less serious cost.

In this arrangement of instructional exercises, we will find out about this MSP430G2 LaunchPad and how to program it. Utilizing this LaunchPad we can work with MSP430 Microcontrollers which offers 16-piece execution with an operational accelerate to 16MHz. The instructional exercises are composed for very tenderfoots in gadgets and henceforth every point would be advised as fresh as could be awaited under the condition. The equipment required for these instructional exercises would be an ordinary PC and the MSP430 Value Line LaunchPad Development Toolkit with scarcely any other essential gadgets segments that you can quit a bit of a stretch find in your nearby hardware handyman store. So with no further ado how about we drop into the Development Tool and look at what's remembered for the case and how to utilize them. We will have the option to Blink a LED toward the finish of this instructional exercise.

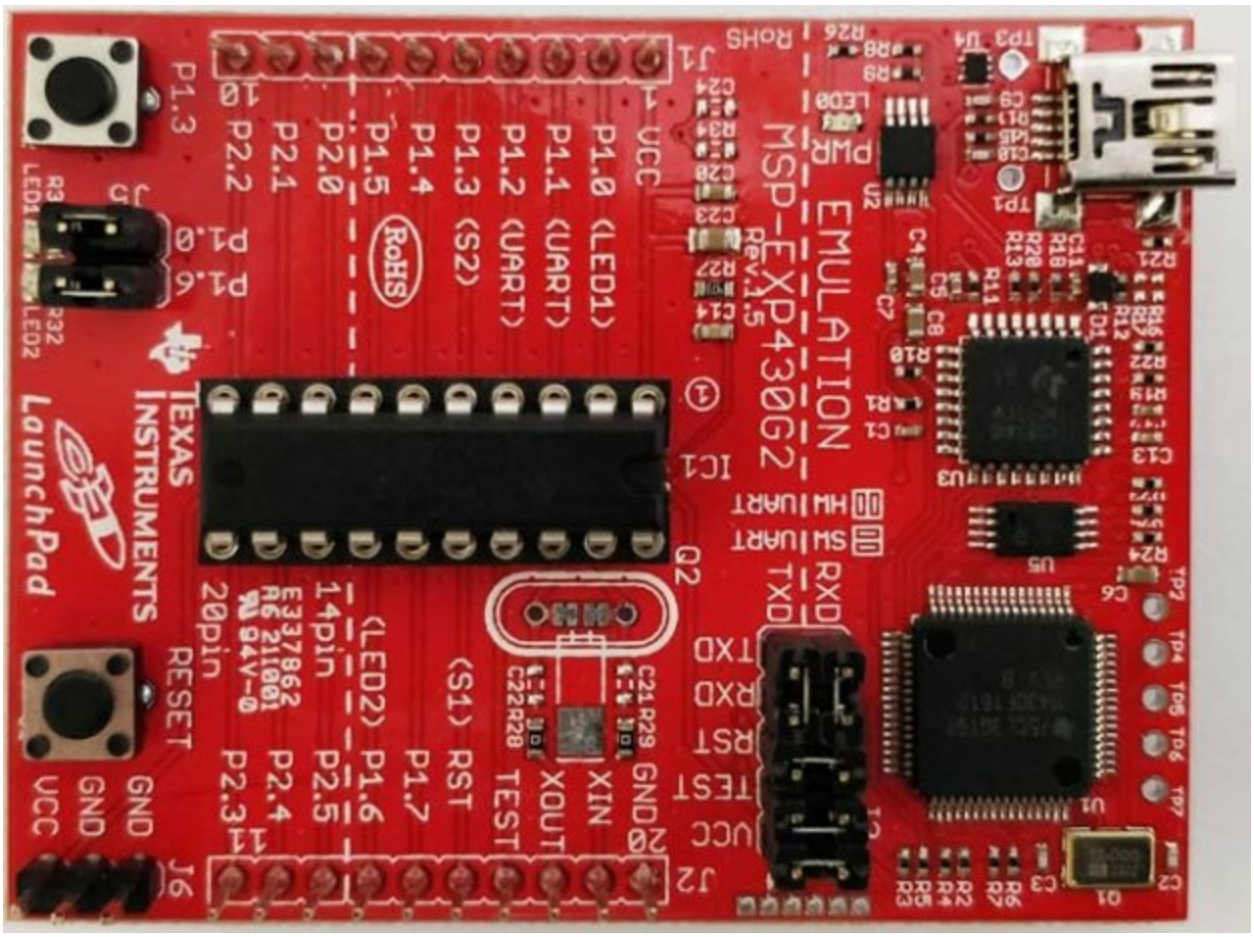
MSP430G2 LaunchPad Contents:

At the point when you buy the MSP430G2 LaunchPad Development Tool from TI or some other nearby seller you will get the accompanying materials remembered for your Box. The total substance are additionally appeared in the image beneath. Likewise note this is relevant as on 2018 the prior and future adaptations may have various Contents

- MSP-EXP430G2 Development Board
- MSP430G2452 along with MSP430G2553 Microcontrollers
- Smaller than expected USB link
- Miniaturized scale Crystal Oscillator (32.7kHz)
- Brisk Start Guide

MSP-EXP430G2 Development Board:

The perfect red shading board is the MSP-EXP430G2 Development Board. This board can program TI Microcontrollers that fall under the MSP430 arrangement. The primary motivation behind this board it to transfer code from the PC to the MCU and read sequential information from the MCU for investigating reason. It likewise gives the pin-out to each stick of the MCU and furthermore two LEDs and a press catch to make improvement simple. The board has developed a ton since its dispatch and the one appeared underneath is the MSP_EXP430G2 Rev1.5. To think about the capacity of every jumper and catch on the board.



MSP430G2452 along with MSP430G2553 Microcontrollers:

As told before the MSP430 Development Board can be utilized to program Microcontrollers that fall under the MSP430 Value line arrangement. In case, with this improvement unit, TI gives us two Microcontrollers from the MSP430 arrangement which are the MSP430G2452 along with the MSP430G2553. Both are 20 Pin DIP IC's with average execution. Naturally, the MSP430G2553 will be fixed to the IC attachment of your Growth board and the MSP430G2452 will be given independently. The specialized detail of both the MCUs are arranged beneath

MCU Name:	Technical Specification
	16kB Flash, 512B RAM, 16GPIO, 2×16-bit Timer, Watch

MSP430G2553	Dog Timer, Brown Out Reset, 1×USI(IIC/SPI/UART), 8ch 10-bit ADC, 8ch Comparator, Capacitive Touch IO Module
MSP430G2452	8kB Flash, 256B RAM, 16GPIO, 1×16-bit Timer, Watch Dog Timer, Brown Out Reset, 1×USI(IIC/SPI), 8ch 10-bit ADC, 8ch Comparator, Capacitive Touch IO Module

As should be obvious the MSP430G2553 has preferable details over the other, it additionally has a UART module which would be helpful while troubleshooting utilizing Energia. Consequently in this arrangement of instructional exercises, we will utilize the MSP430G2553 to investigate all the functionalities of this Development unit.

Scaled down USB link:

The small USB link is utilized to associate the board the Computer when a program has transferred the information (in type of hex code) will spill out of the PC to the board through this link. Likewise during troubleshooting (Serial Monitor), the information from the MCU will be gotten by means of this link.

This link additionally gives capacity to the board, so you can even utilize a versatile charger to control your board through this link in the wake of transferring the program.

Small scale Crystal Oscillator:

TI Also gives a 32kHz small scale precious stone Oscillator alongside the Development Kit. This Crystal can be fastened to the Board yet it is totally discretionary. Since the MSP430G2553 IC has an interior Oscillator of 16MHz which must be enough for us to begin.

Controlling and Testing your Development Board:

Before we begin anything TI would have just transferred an example

Program on your MSP430G2553 Microcontroller, so let us power the board and check in case it is working. You can control board through the small USB jack and once you do it, you should view the LEDs (red and green) at the base left corner of your board shining on the other hand. You would then be able to press the press button associated with P1.3 to check if the inner temperature sensor is working. Truly, the MSP2553 has an inner temperature sensor, subsequent to squeezing the catch simply rub your fingers to warm it up and place it on the IC you can view the Red LED goes on to demonstrate the ascent in temperature. Cool!! Right?? OK presently, let us proceed onward to the Software Environment.

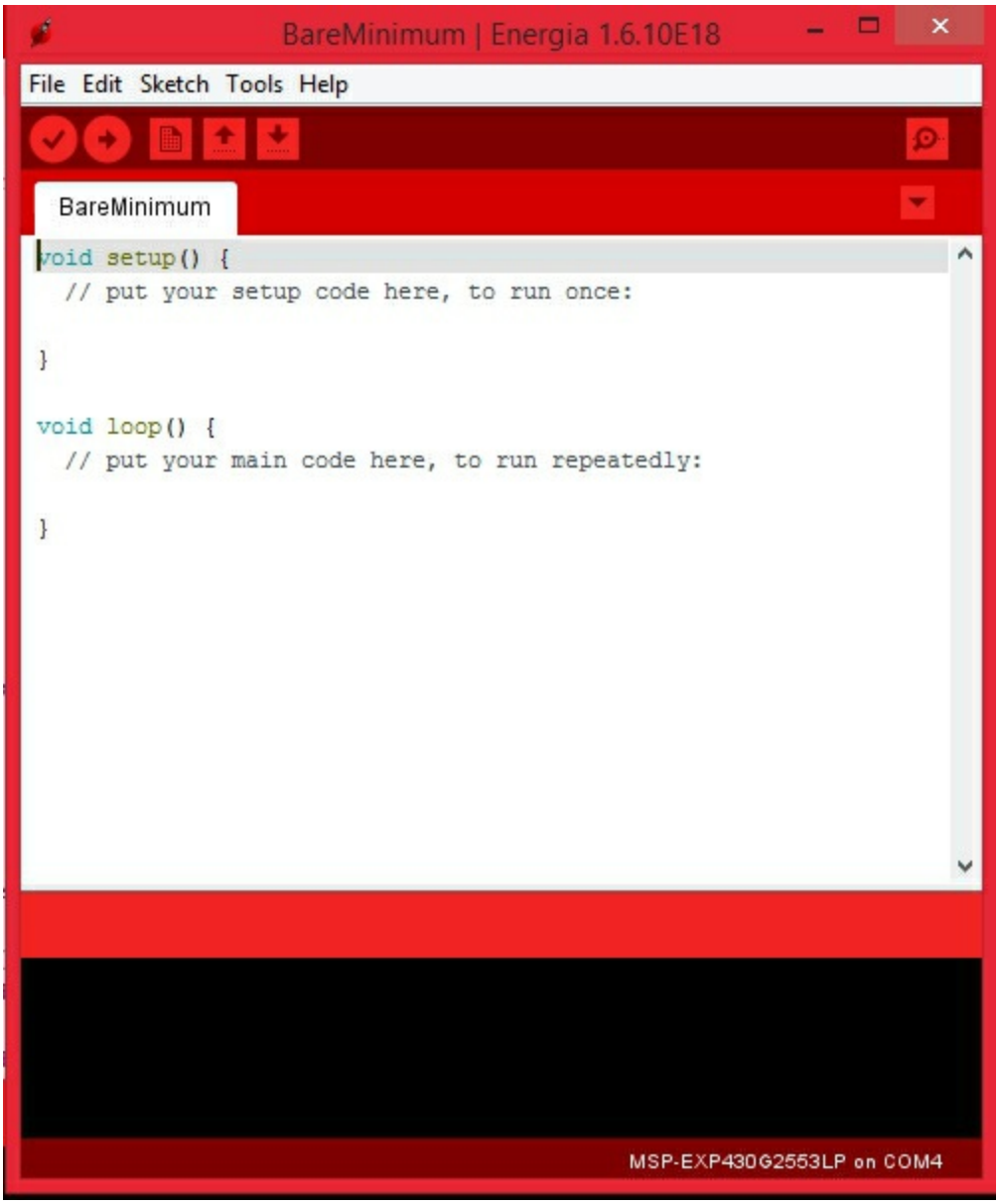
Programming Software (IDE) for MSP430 LaunchPad:

Texas Instruments permits us to program their Microcontrollers across an assortment of Environments. The Official 1 is the Code Composer Studio usually known as the CCS. This product is additionally free however utilizing it is requires some insignificant degree of involvement in Microcontrollers.

Since this arrangement of instructional exercises are focused for supreme fledglings we utilize another Development Environment called Energia. Energia is an Free access along with free Environment that empowers us to program the TI Microcontrollers without any problem. The primary point of Energia is to do programming TI MCU's as simple as programming in Arduino. So Energia is an Equivalent for Arduino that underpins Texas Instruments Microcontrollers. Individuals who have utilized Arduino will concur more on this once they download and dispatch the Energia IDE.

Downloading and propelling the Energia IDE:

As said before the Energia is an free access along with free Development condition, and it very well may be downloaded from this Download interface. Select the Version dependent on your working framework, for windows you should see a ZIP record being downloaded. In the wake of downloading the ZIP simply separate it on your favored area and open the organizer. You discover the application named Energia. Dispatch it and it should Look like underneath.



Arduino clients don't be amazed as it could very well resemble your Arduino IDE has spruced up like a Santa to show up as Energia. The likeness of both Arduino and Energia are same since the two of them are based on a stage called Processing.

Flickering a LED on MSP430G2:

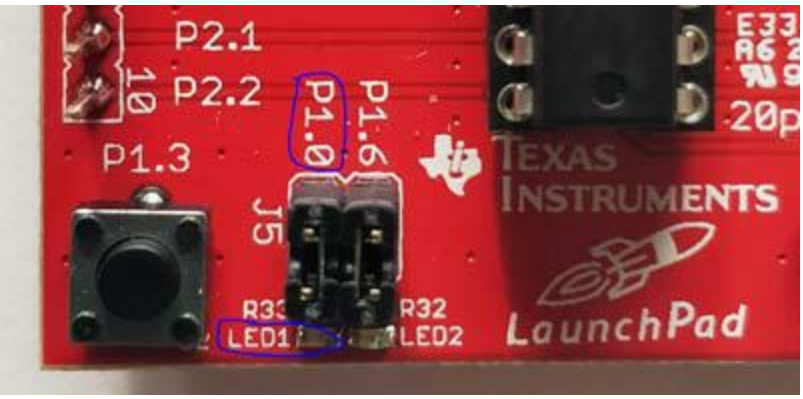
Presently, that we are prepared with our equipment and programming let us attempt an essential model program from Energia to squint the on board

LED. Before we begin programming we require to realize the pin names of each pin on our MSPG2553 IC. Since we will utilize these names while programming our board. The accompanying picture from the Energia site will assist us with understanding the name and usefulness of each pin. In view of the Revision of your board the picture may fluctuate marginally.

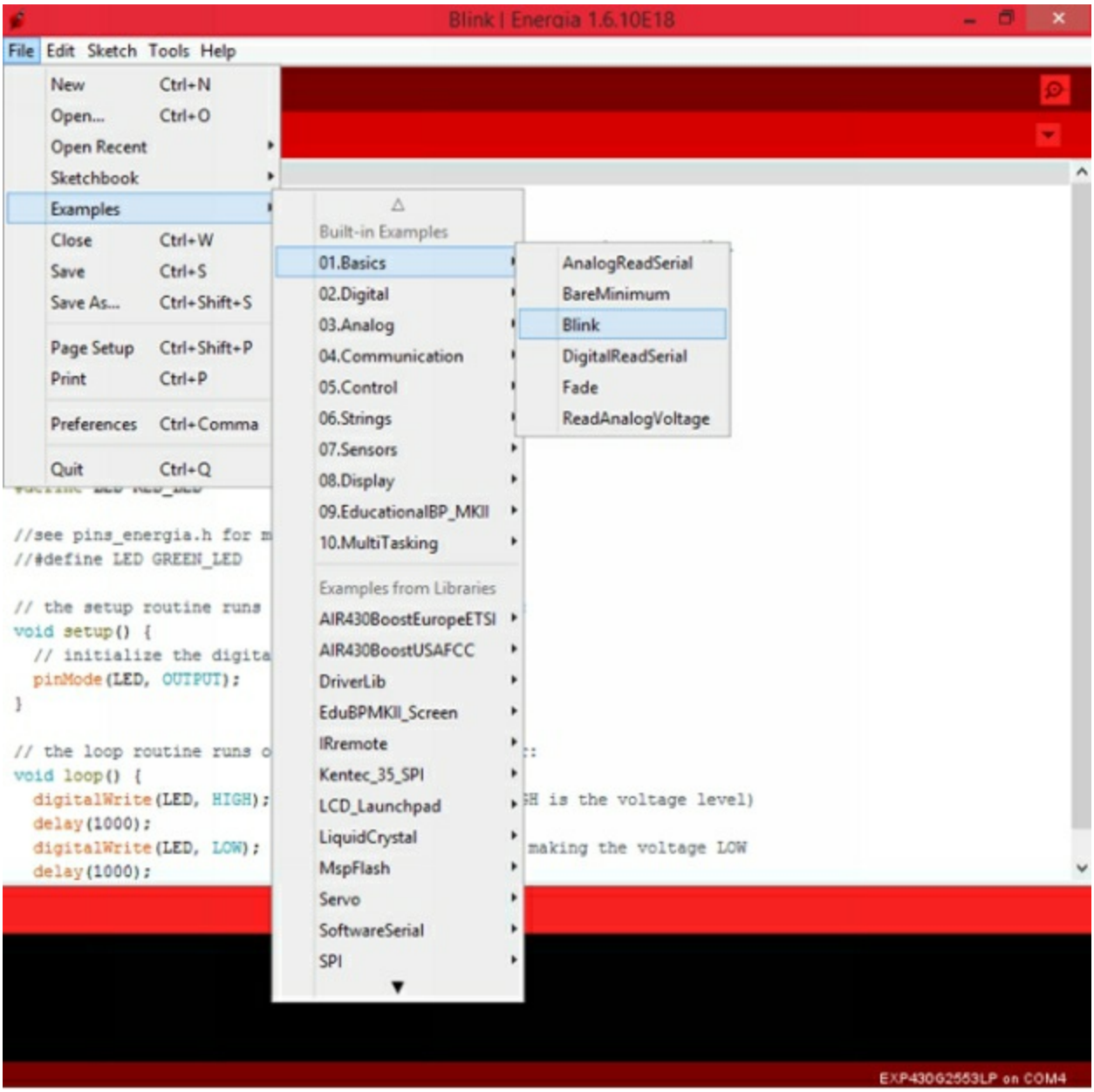
Spare this picture, as we will require everything time while programming our MSP430 through Energia.

Understanding the Example Blink Program:

Let's, start with the Example squint program which will flicker the LED1 (red shading) which is associated with the P1.0 pin of your Microcontroller. The Light Emitting Diode along with its stamping on my board is in the beneath picture.



To open this model program explore to File->Example->Basics-> Blink as demonstrated as follows



The accompanying system will show up on your IDE

```
#define LED RED_LED

// the setup routine runs once when you press reset:

void setup() {
```

```
  pinMode(LED, OUTPUT);     // initialize the digital pin as an output.

}

// the loop routine runs over and over again forever:

void loop() {

  digitalWrite(LED, HIGH);    // turn the LED on (HIGH is the voltage level)

  delay(1000);              // wait for a second

  digitalWrite(LED, LOW);     // turn the LED off by making the voltage LOW

  delay(1000);              // wait for a second

}
```

Lets break the above code into line by line and see what it really implies, yet before that let us comprehend the fundamental programming structure of Energia. Each Energia Program will have two compulsory capacities, they are void Setup() and void Loop().

The code present inside the void Setup() will be executed just a single time and the program present inside the void Loop() will be executed for ever. All the pin presentations and initialisations will be done inside the Setup() and the fundamental program which must be executed perpetually will be composed inside the void Loop(). You can consider void Loop() as a comparable to while(1) (Infinite while circle). In light of this let us begin investigating above code line by line.

The principal line of the program is called macros. The Energia IDE is sufficiently brilliant to comprehend words like RED_LED, GREEN_LED, TEMP_SENSOR and substantially more. These are only the name of the pins to which the equipment is associated with. This is conceivable on the grounds

that the Red and Green Led's are designed on our board and subsequently the MCU pin to which it's associated with is known. Again for the solace of programming we utilize a Macros which says that as opposed to utilizing the name "RED_LED" in my program I will utilize just "Drove" by utilizing the #define as appeared in the underneath line.

```
#define LED RED_LED
```

Next we come into our void arrangement() work, this is where we tell our MCU which pins ought to be utilized as information pins and which ought to be use as yield pins. In our program we just utilize a LED and it is an Output gadget, so we announce it to be a yield pin like demonstrated as follows

```
void setup() {

  pinMode(LED, OUTPUT);     // initialize the digital pin as an output.

}
```

The line pinMode() is utilized to state that I will announce input/yield pins and afterward we state the name and kind of the pin in section. Here the name of the pin is LED and the kind of the pin OUTPUT.

There are numerous manners by which we can call a pin. In this model we have named it LED utilizing the #define large scale, however we can likewise name it utilizing its unique name. Like for our situation the Light Emitting Diode is associated with the second pin of the MCU and that pin is named as P1.0, Refer the pin-out picture above to see the names. So as opposed to calling it as LED we can likewise call it by the accompanying

```
pinMode(LED, OUTPUT);     //Instead of this we can also

pinMode(2, OUTPUT);     //Call by pin number
```

```
pinMode(P1.0, OUTPUT);     //Call by pin name
```

Next lets step down to the void Loop() work. Here we require to compose the code to squint the LED. To squint a LED we require to turn it ON sit tight for a pre-characterized coordinated and afterward turn it OFF and again hang tight for a pre-characterized on numerous occasions the cycle proceeds.

To kill a pin ON or in Energia we require to utilize the digitalWrite() work. The parameters pass the name of the pin and its state in the sections like demonstrated as follows

```
digitalWrite(LED, HIGH);     // turn the LED on (HIGH is the voltage level)
```

Here the name of the pin is LED and the state is HIGH, likewise you can likewise turn it off by utilizing the state LOW like underneath

```
digitalWrite(LED, LOW);     // turn the LED off by making the voltage LOW
```

As said before the name of the pin can be in any way similar to LED, 2 or P1.0 for this pin. So different structures like

```
digitalWrite(2, LOW); digitalWrite(P1.0, LOW);
```

are likewise conceivable.

Presently, that we have find out how to kill ON or the pins. We will figure out how to present defer utilizing the postponement(). We can determine delay by passing an incentive inside the sections as far as milli-seconds

```
delay(1000);              // wait for a second
```

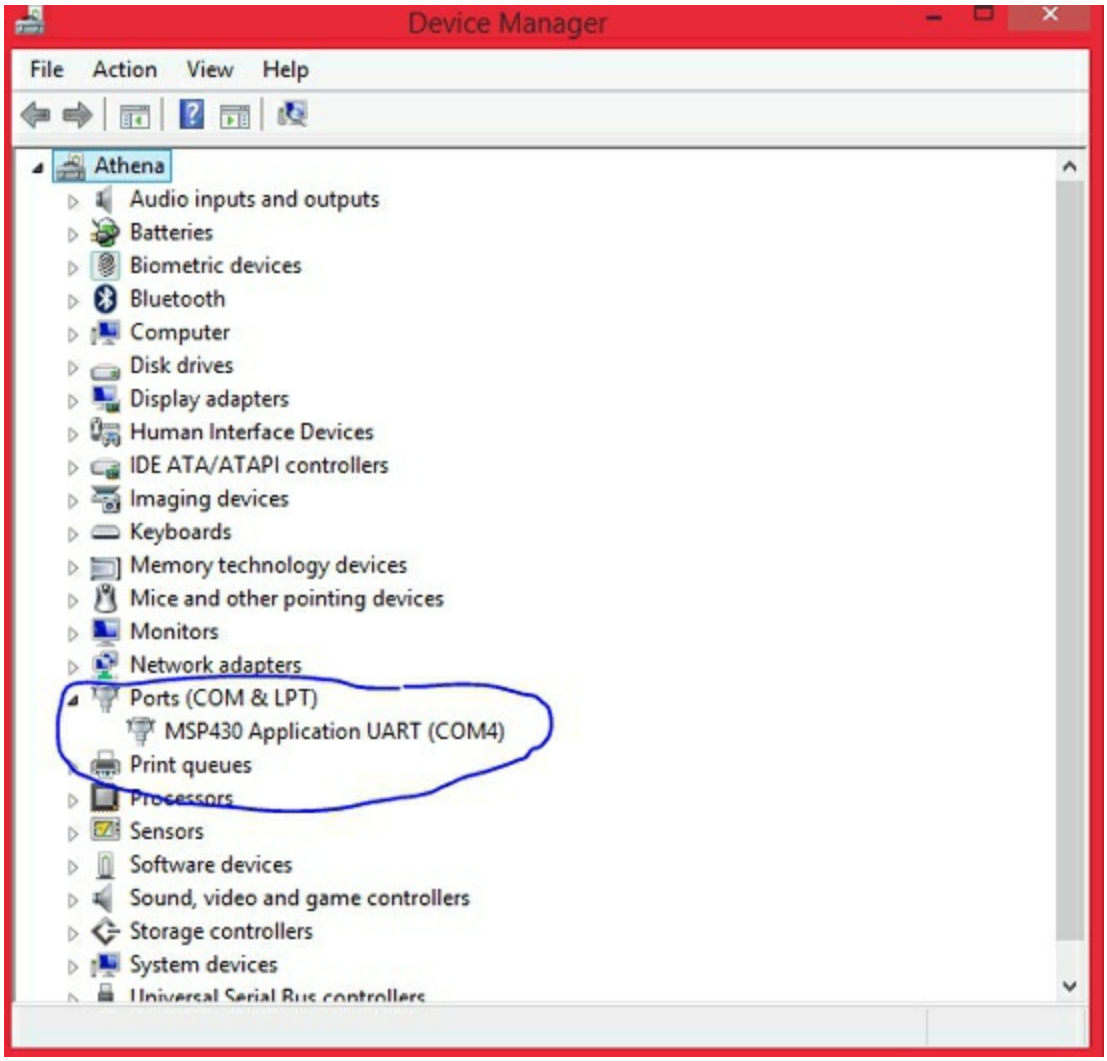
So how about we consolidate all these into our while circle. We should turn

ON the LED hang tight for 1sec, at that point turn it OFF and again sit tight for 1sec. This cycle should proceed until the end of time. So our program will look like this beneath

```
void loop() {

  digitalWrite(LED, HIGH);    // turn the LED on (HIGH is the voltage level)

  delay(1000);              // wait for a second

  digitalWrite(LED, LOW);     // turn the LED off by making the voltage LOW

  delay(1000);              // wait for a second

}
```

Incorporating and transferring your Blink Program:

The following stage is transfer this program to our MSP board. To do this just interface your load up to the PC utilizing the USB smaller than normal link gave and hold on to some time. The drivers for the board should initiate introducing consequently. At that point Open your gadget administrator and under the COM ports alternative you must view your Board name like demonstrated as follows



In the event that you don't discover the board found, go to the underneath Energia driver's connection and introduce the proper drivers.

- Windows
- Macintosh OS X
- Linux

When your board is found, make a note to which COM port it is associated with. Mine is associated with Port 4 here. At that point return to the Energia IDE along with select Tools - > Port along with choose a similar port I have select COM4 for me. Simultaneously go to Tools - > Boards and select the MSP-EXP430G2553LP. When done you should see the accompanying at the

base right corner of your Energia IDE.

Presently click on the transfer symbol on the upper left corner and your program should begin transferring to your board. In case everything works fine you should see the "Done transferring" message showing up on your IDE as demonstrated as follows

Assuming else, you may get the regular blunder called "unfit to discover a gadget coordinating 0451:f432". Follow the subsequent stage in case you get this mistake, else continue to checking the yield.

Unfit to discover a gadget coordinating 0451:f432 [Solution]:

This Error is basic with the Energia IDE however fortunately there is a changeless answer for this that could be found at this discussion string.

I you are not a major aficionado of recordings at that point basically follow the means beneath.

- duplicate <energia directory>\hardware\tools\DSLite\DebugServer\drivers\MSP430.dl to <energia directory>\hardware\tools\msp430\bin\

- alter <energia directory>\hardware\energia\msp430\boards.txt a change the 2 events of rf2500 with tilib

- restart energia along with you ought to have the option to transfer to the MSP-EXP430G2 with MSP430G2553.

Confirming our Blink Output:

When the program is transferred you can see the Red LED on your Board squinting with a postponement of 1000mS (1 sec) as modified from our Energia IDE as demonstrated as follows.

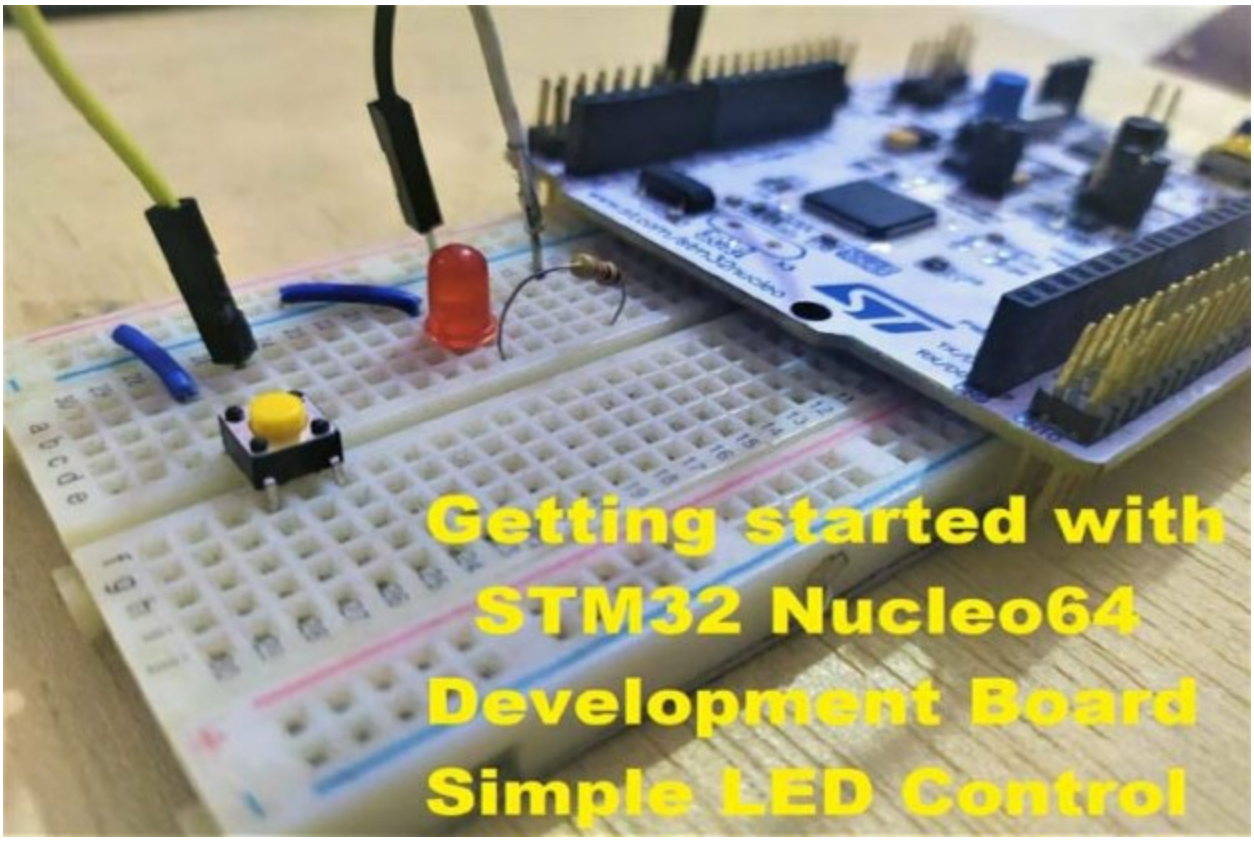
In case everything has worked out till now, at that point give yourself a treat and rigging up for the following instructional exercise for all the more energizing stuff.

Expectation you comprehended the instructional exercise and mastered something valuable.

Code

```
#define LED RED_LED

// the setup routine runs once when you press reset:

void setup() {

  pinMode(LED, OUTPUT);     // initialize the digital pin as an output.

}

// the loop routine runs over and over again forever:

void loop() {

  digitalWrite(LED, HIGH);    // turn the LED on (HIGH is the voltage level)

  delay(1000);              // wait for a second

  digitalWrite(LED, LOW);     // turn the LED off by making the voltage LOW

  delay(1000);              // wait for a second

}
```

3. BEGINNING WITH STM32 NUCLEO64 UTILIZING STM32CUBEMX AND TRUESTUDIO - SIMPLE LED CONTROL



A significant number of us ought to be acquainted with the famous microcontrollers and improvement sheets like Arduino, Raspberry Pi, ESP8266, NoduMCU, 8051, along with etc. Indeed, for the vast majority, Arduino would have been their first advancement board, yet as we burrow profound and start proficient plans, we will before long understand the confinements of Arduino (like cost, flexibility, strength, speed, along with etc.) and comprehend the need to move into an increasingly local Microcontroller stage like PIC, STM, Renesas, along with etc.

We have just secured a succession of PIC Microcontroller instructional exercises, which guides fledglings for learning PIC microcontrollers. Likewise, beginning with this article, we will likewise design a grouping of STM32 Nucleo64 Development Board Tutorials which can assist outright novices with learning and create utilizing the STM32 Platform. Nucleo64 Development Boards are ease and simple to utilize stage for proficient engineers just as for specialist. In case you are totally new to the STM32 Nucleo64 Development Boards, do look at this Nucleo64 Review to comprehend the essentials of this board before you continue further. The

likewise exhibits how to program STM32 utilizing ARM Mbed Platform yet for this instructional exercise, we will utilize another allowed to utilize stage from ST Microelectronics called TrueSTUDIO.

Note: There are numerous adaptations of STM32 Nucleo64 Development Boards, the specific board utilized in this instructional exercise is NUCLEO-F030R8. We have chosen this board chiefly in view of its minimal effort. Indeed, in case you have an alternate form, most things talked about in the instructional exercise will get the job done for you to begin.

Choosing and Downloading the Required Development Platforms for Nucleo64 Boards

Beginning with any microcontroller will require a programming IDE like we have Arduino IDE for Arduino sheets, Atmel Studio for AVR microcontroller, MP Lab for PIC, and so on. So here we additionally need an IDE for our STM32 Nucleo64 Boards to perform programming and troubleshooting. The STM32 family comprises of 32-piece Microcontrollers that help the accompanying IDEs and toolchains:

- IAR Embedded Workbench® for ARM® (EWARM).
- MDK-ARM Keil
- TrueSTUDIO
- Framework Workbench for STM32

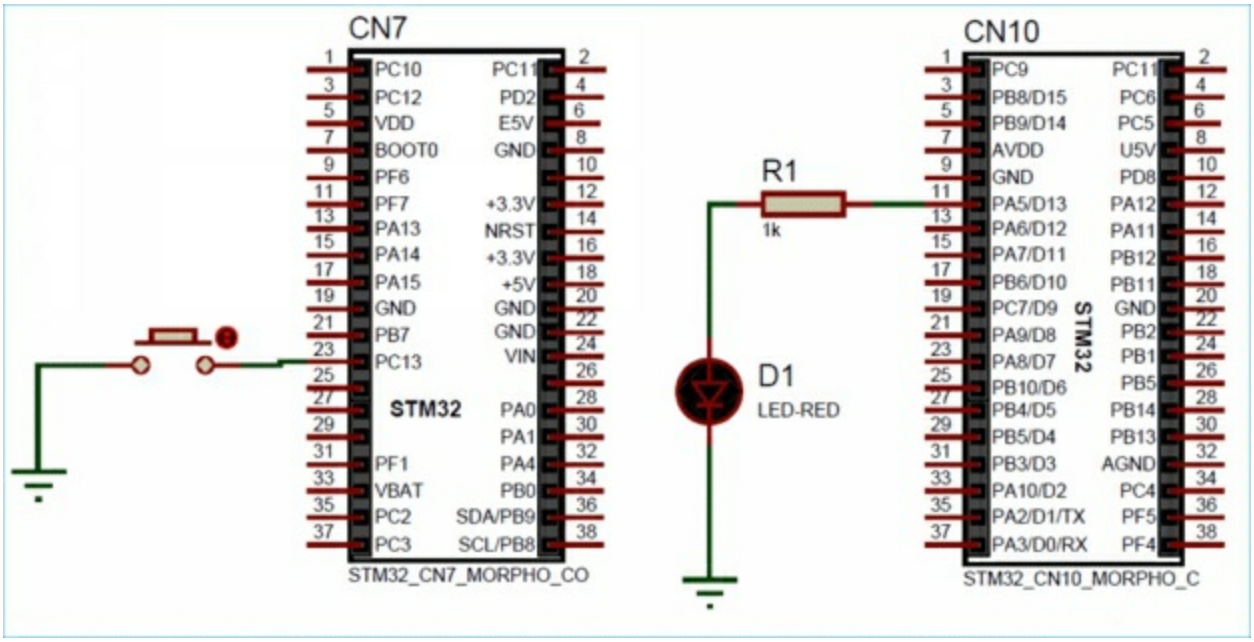
Here for our instructional exercises, TrueSTUDIO will be utilized for composing, ordering, and troubleshooting code since it is allowed to download and utilize in any event, for business ventures with no permit prerequisite. At that point STM32CubeMX will be utilized to produce fringe drivers for STM32 sheets to make programming simple. To transfer our program (hex record) into our improvement board, individuals ordinarily utilize the STM32 ST-LINK Utility apparatus, yet rather, we will utilize TrueSTUDIO itself to do this. TrueSTUDIO has an investigate mode that permits software engineers to transfer the hex document straightforwardly to

the STM32 board. Both TrueSTUIO and STM32CubeMX is anything but difficult to download, simply follow the connection beneath, information exchange and download the arrangement. At that point introduce them on your Laptop.

- Download STM32Cube MX
- Download TrueSTUDIO

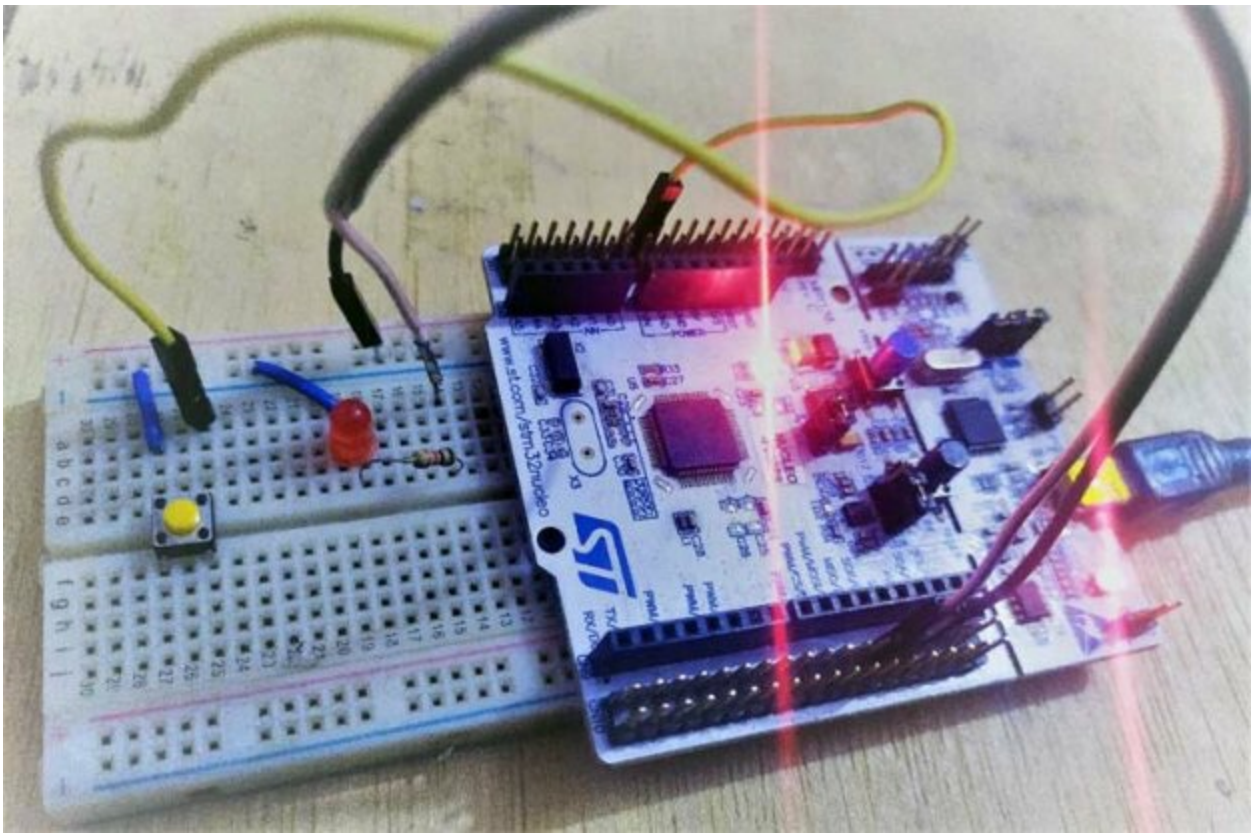
Circuit Diagram and Hardware arrangement

Before we continue with the product area and coding, we should set up our board for this undertaking. As referenced before here, we are gonna to control a LED utilizing a press button. Presently, you should definitely realize that your STM32 Development Board has two arrangements of connector nails to either side called ST Morpho pins. We have associated a press button and a LED to these pins as appeared in the circuit outline underneath.

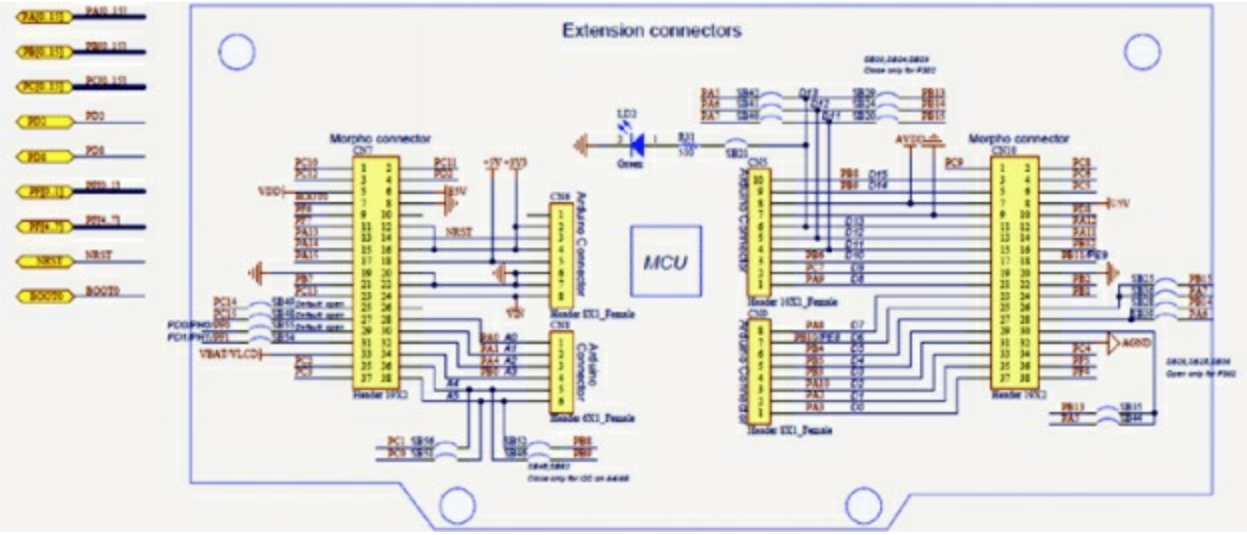


Circuit associations are simple for this venture, we have to interface a LED at PA5 of PORTA and a switch at PC13 of PORTC as for GND. When the

associations were made, my test set-up resembled this.

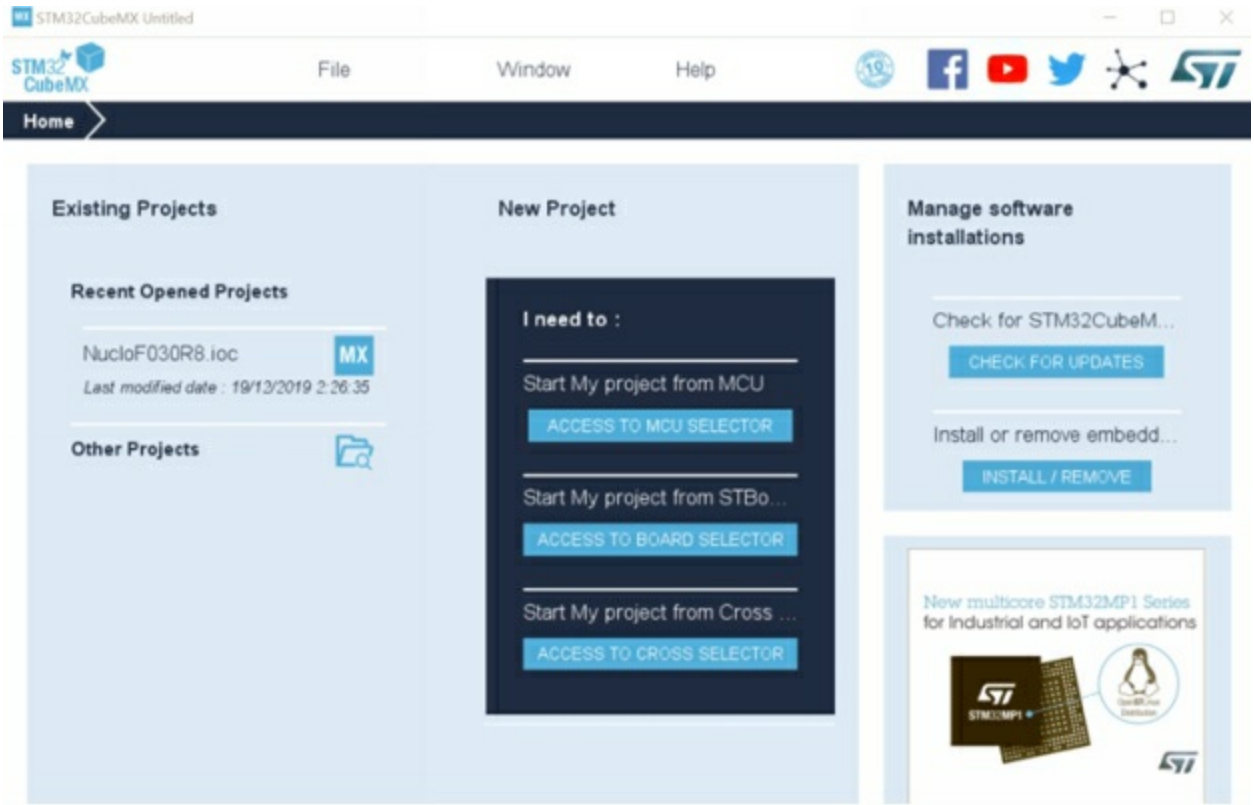


On the other hand, we can likewise utilize the inbuilt LED and press button on the board. These inbuilt LEDs and press button additionally associated at a similar pin as appeared in the circuit graph. We have included outside parts just for training. The beneath pin outline of the STM32 Development Board will prove to be useful to know where each morpho pins are associated with locally available.

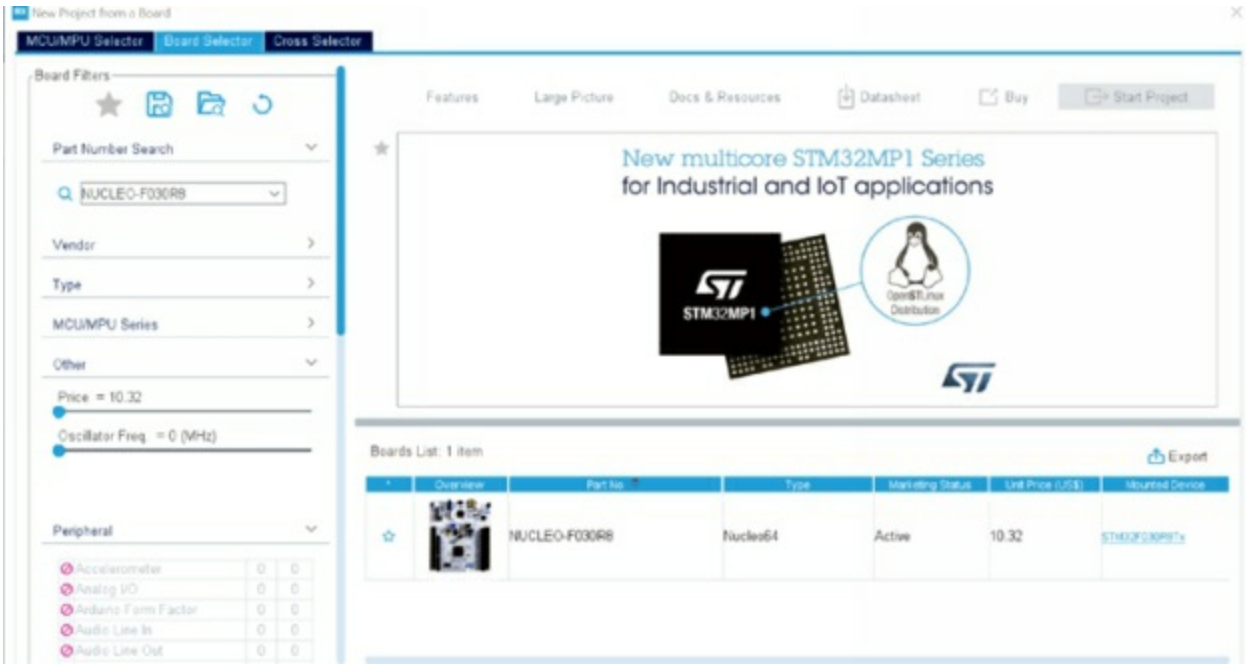


Beginning with STM32CubeMX for STM32 Nucleo64 Development Boards

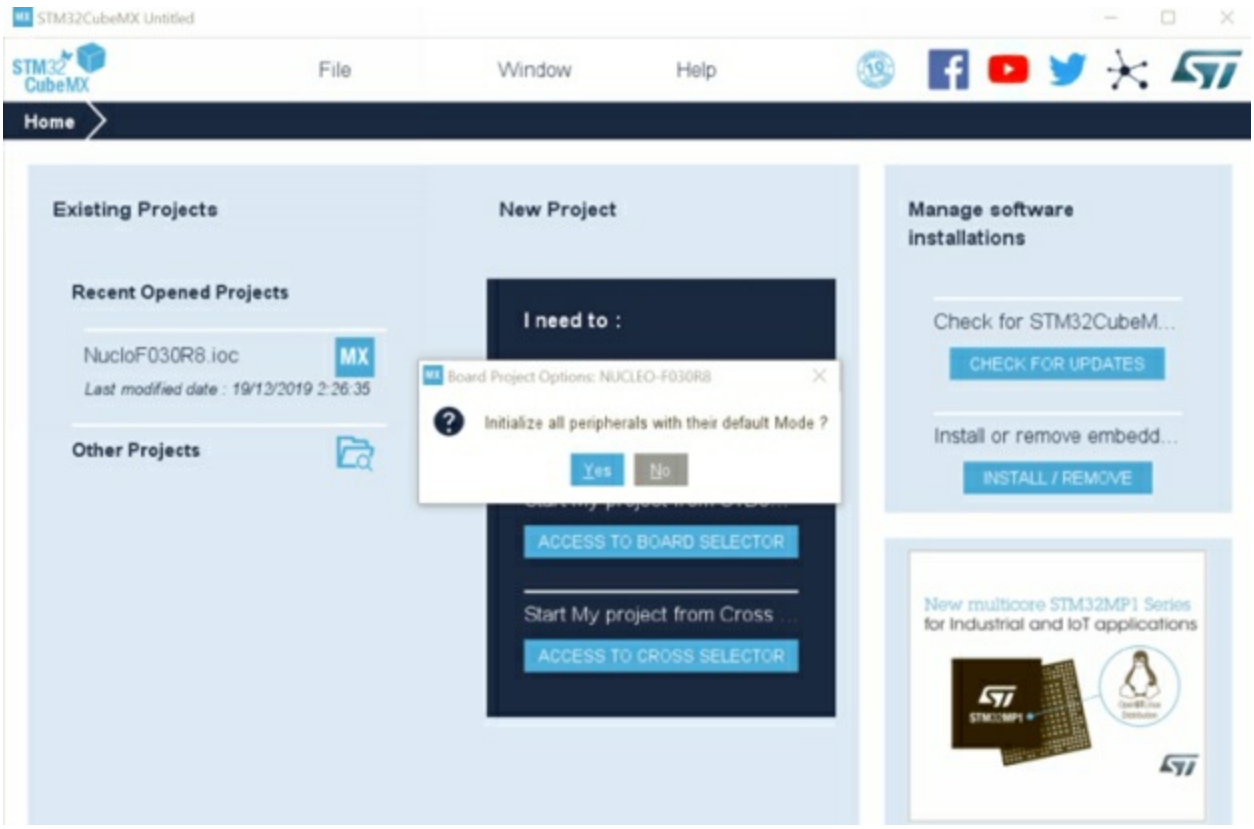
Stage 1: After establishment, dispatch STM32CubeMX, at that point select the entrance board selector to choose the STM32 board.



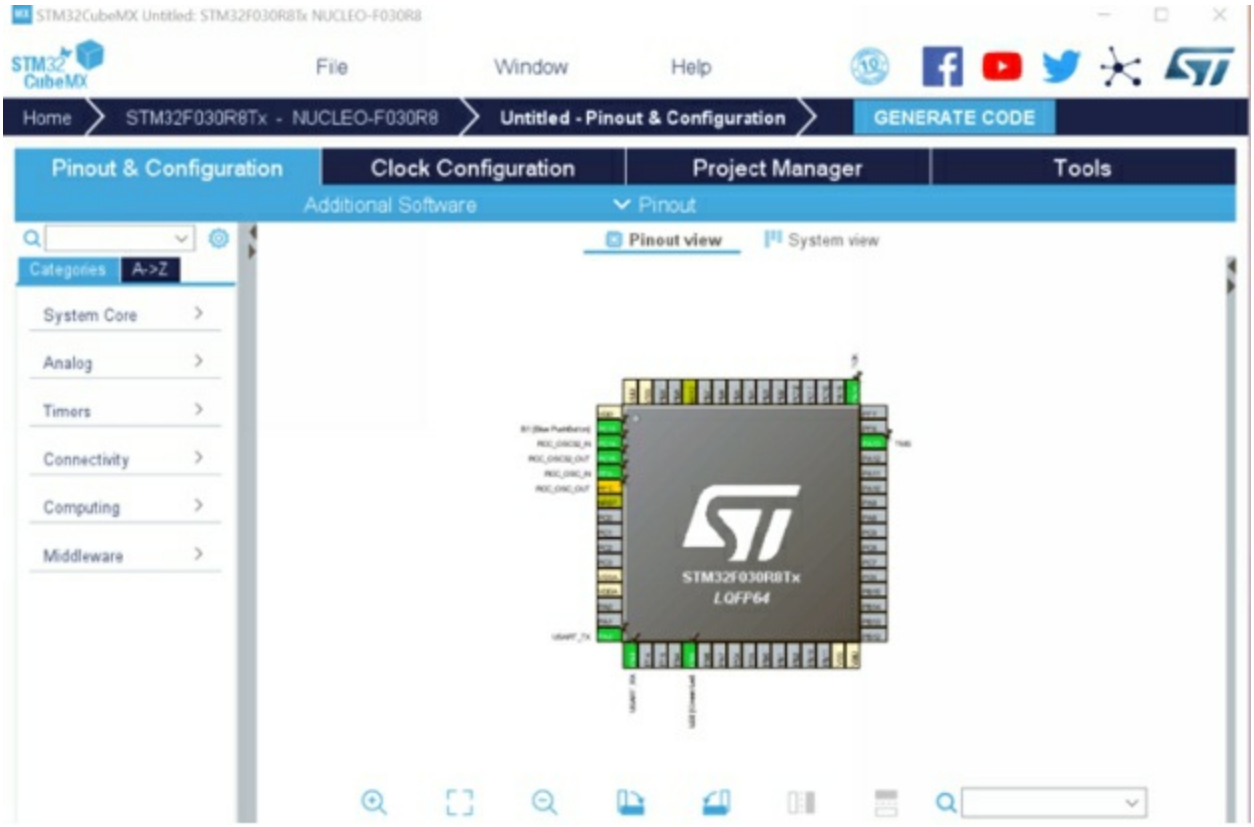
Stage 2: Now search board by your STM32 board name like NUCLEO-F030R8 and snap on the board appearing in the image. In the event that you have an alternate board look for its individual name. The product will bolster all STM32 improvement sheets from ST Microelectronics.



Stage 3: Now click on yes as appeared in the image beneath, to instate all the peripherals in their default mode. We can later change the necessary ones varying by our venture.



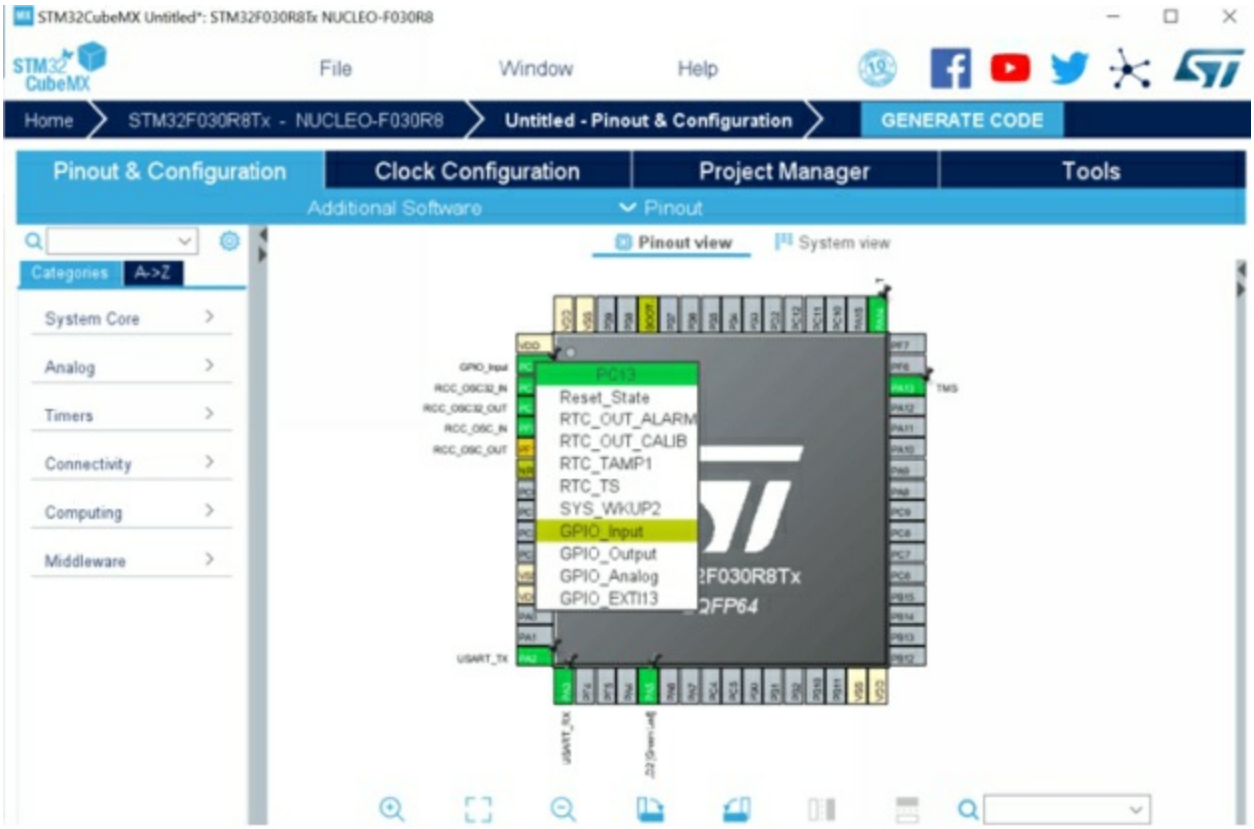
In the wake of tapping on 'Yes', the screen will like the beneath picture and green shading pin demonstrating that they are started of course.



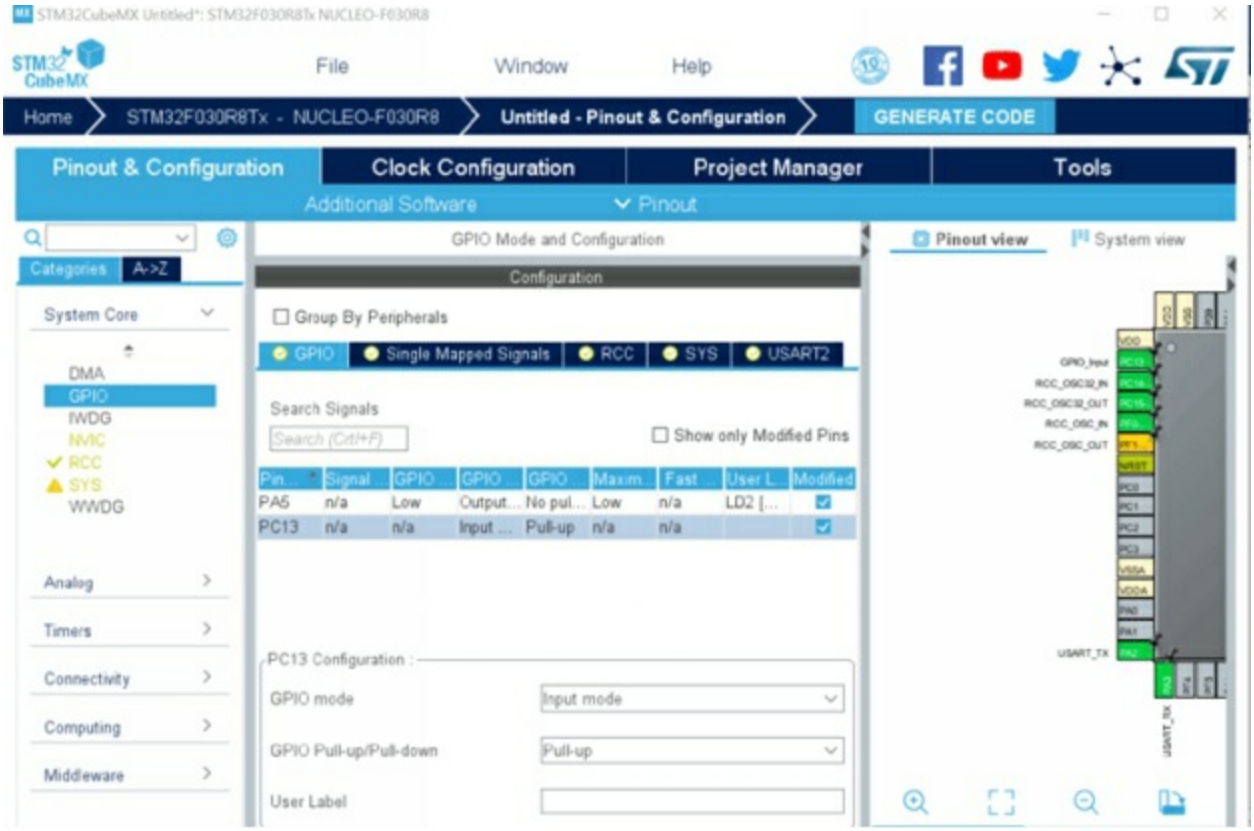
Stage 4: Now clients can choose the ideal setting from the classes. Here in this instructional exercise, we are going to flip a LED utilizing a press button. In this way, we have to make the LED pin as yield and switch pin as INPUT.

You can choose any pin, however I am choosing PA5 and changing its state to GPIO_Output to make it function as a yield pin as appeared in the underneath picture.

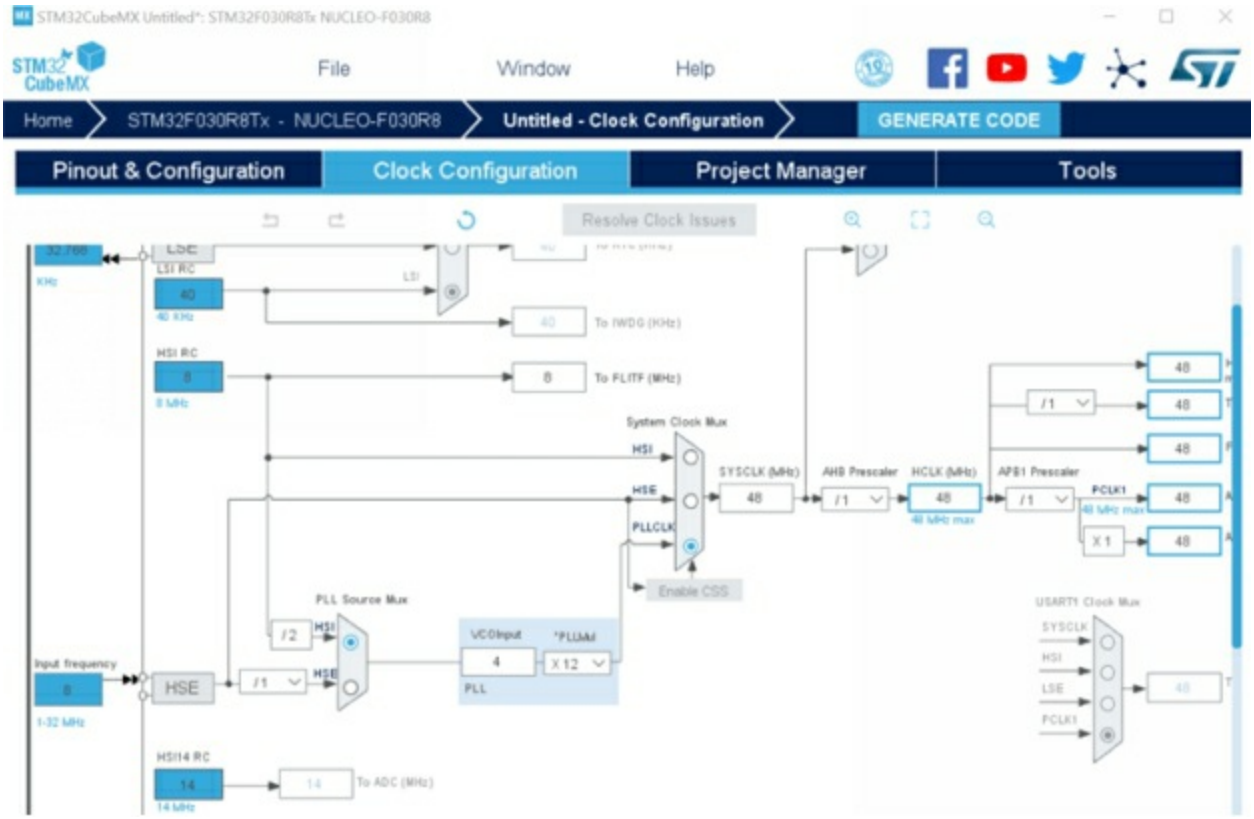
Correspondingly, I am choosing PC13 as GPIO_Input with the goal that I can peruse the status of my press button.



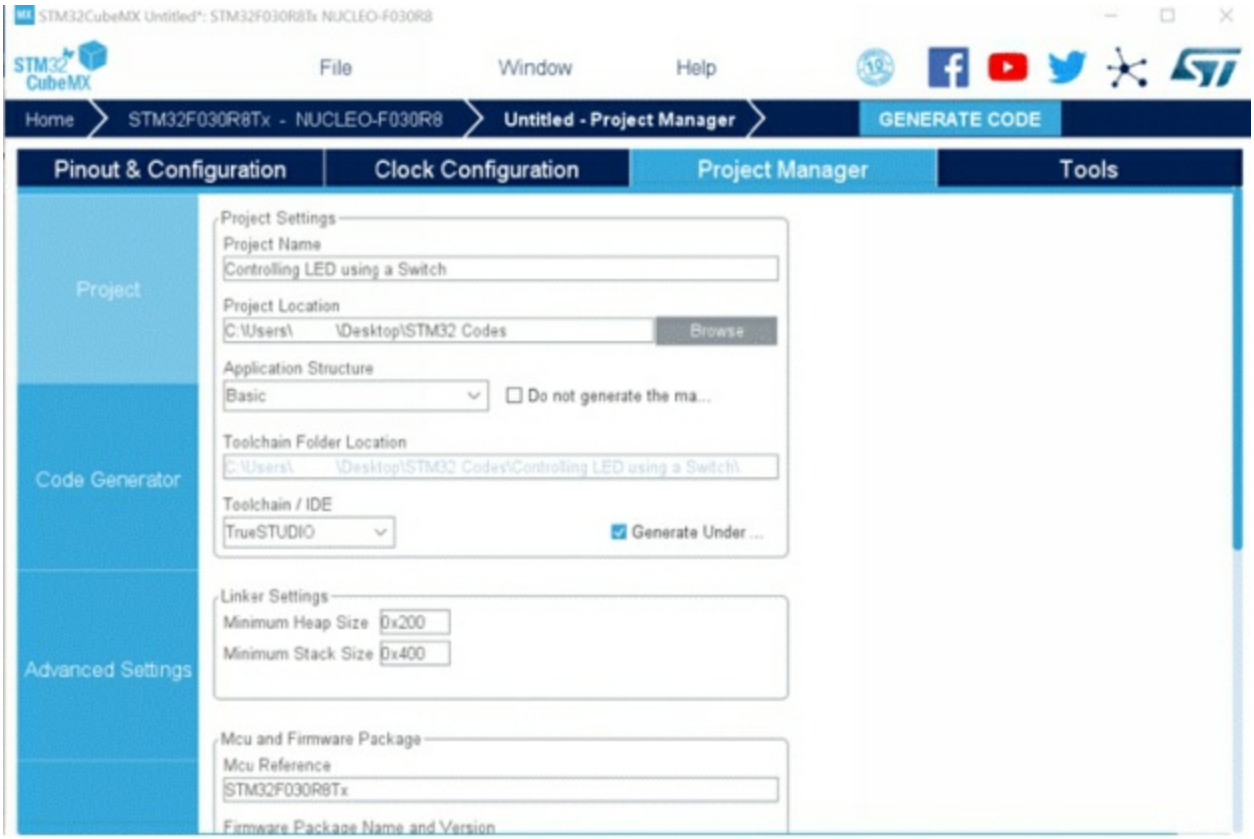
Then again, we can make additionally design pins from the pinout and arrangement tab just as demonstrated as follows.



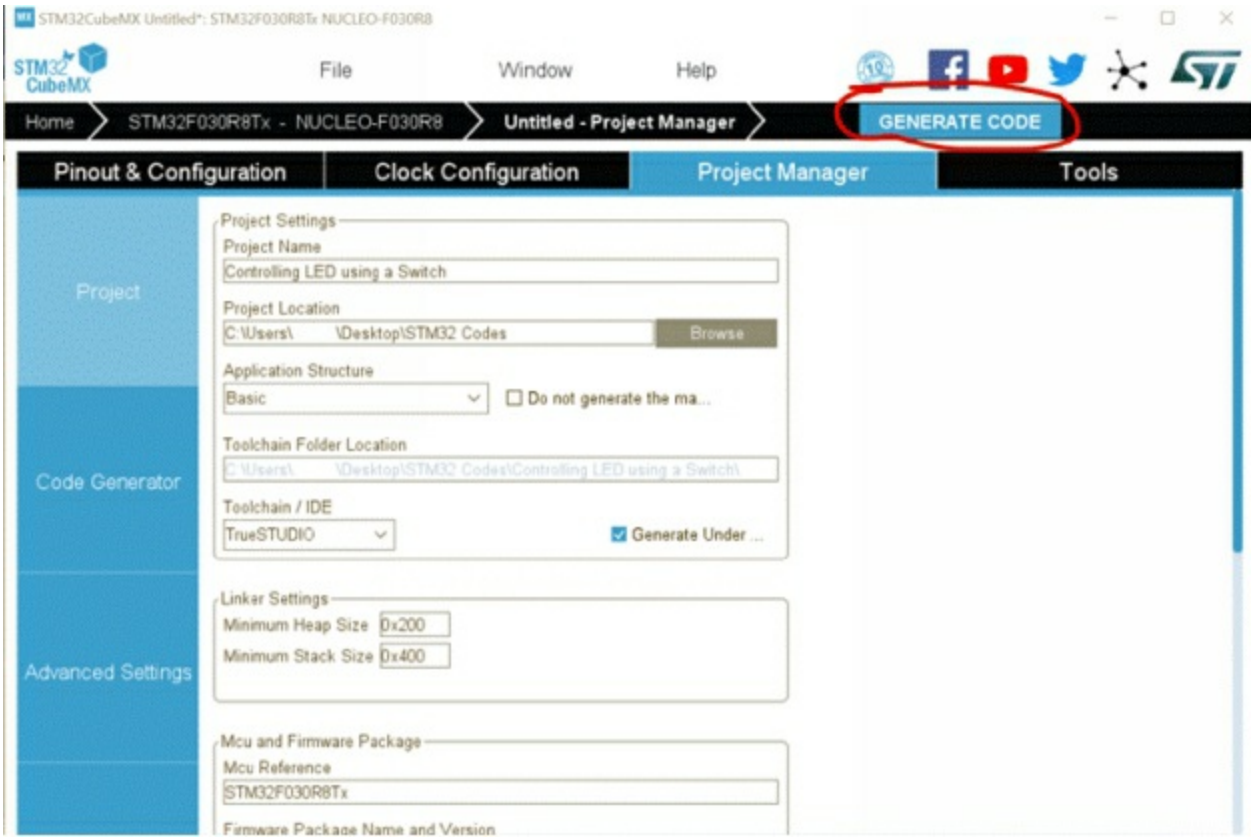
Stage 5: In the following stage, the client can set the ideal recurrence for the microcontroller and pins as indicated by outside and inward oscillator. Of course, an inner 8 MHz precious stone oscillator is chosen and by utilizing PLL, this 8 gets changed over to 48MHz. Which means of course STM32 board or microcontroller and Pins will take a shot at 48MHz.



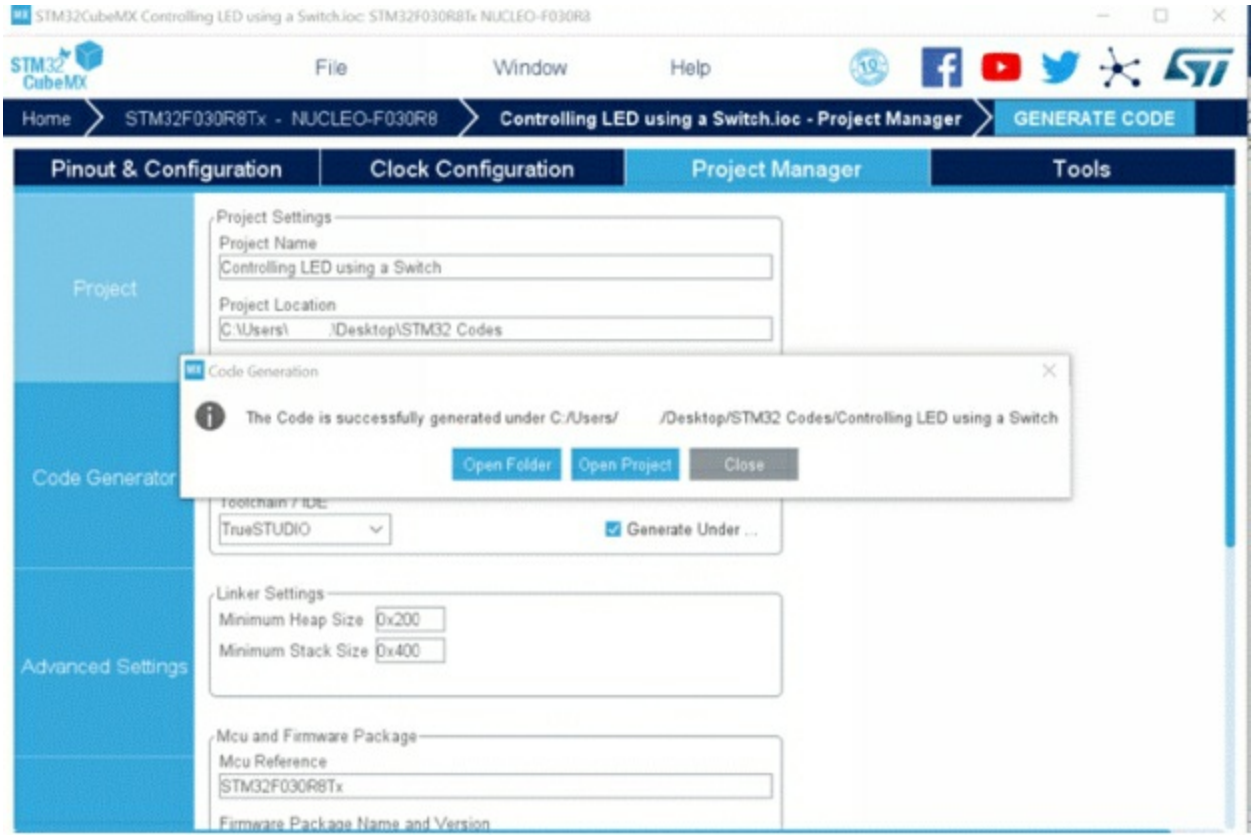
Stage 6: Now move in the undertaking supervisor and give a name to your task, venture area, and select toolchain or IDE. Here we are utilizing TrueSTUDIO, so I have chosen equivalent to demonstrated as follows.



Stage 7: Now click on Generate Code mark by the red hover in the underneath picture.

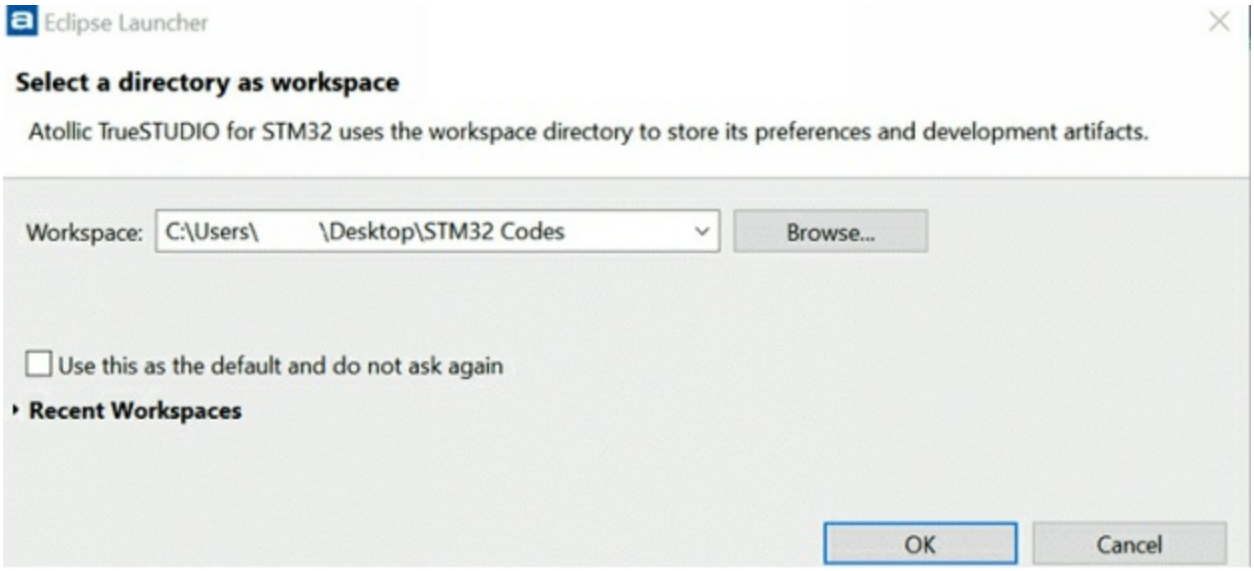


Stage 8: Now you will view a popup as given at that point click on open undertaking. However, ensure you have introduced TrueSTUDIO before this progression.

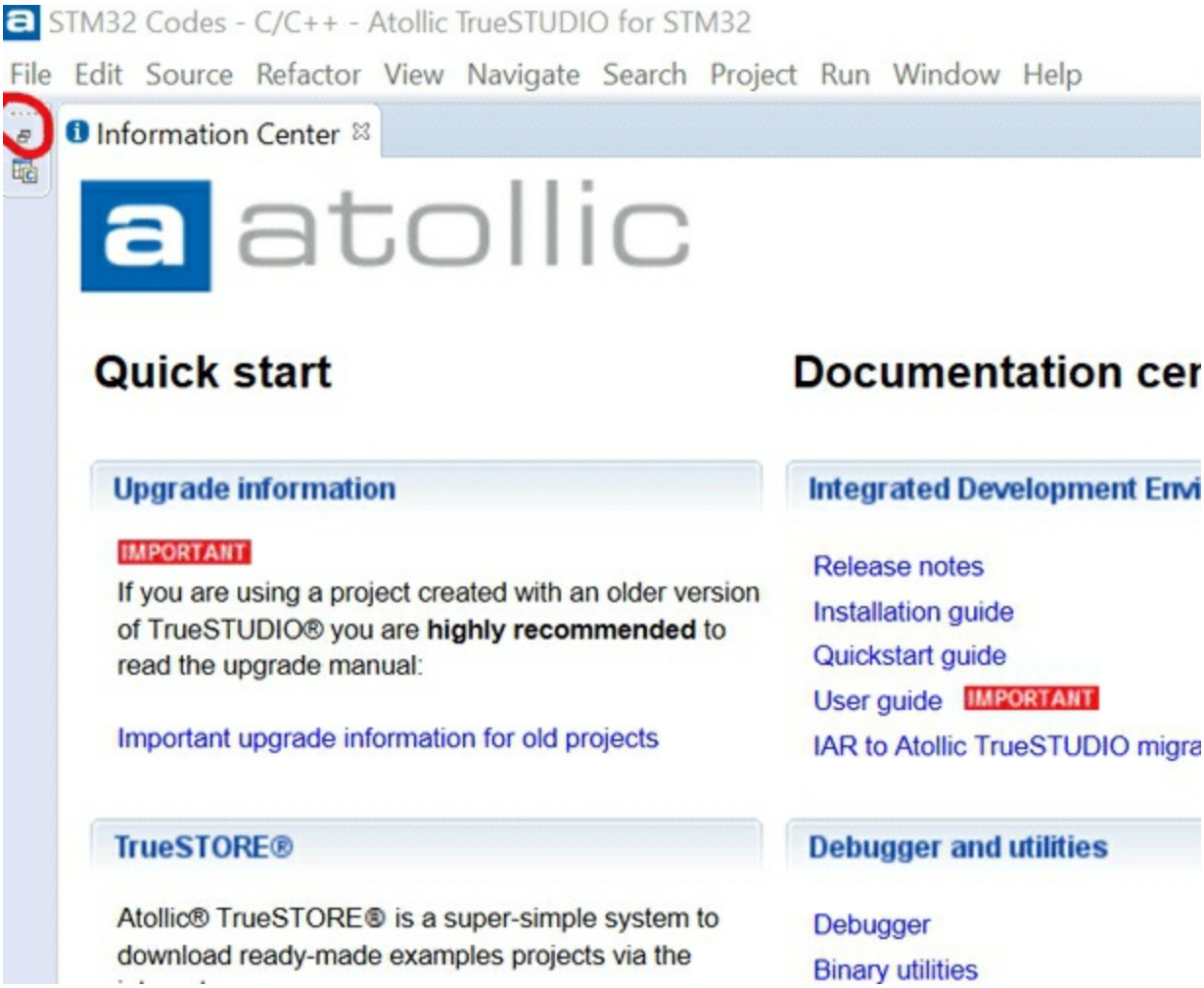


Programming STM32 Nucleo64 Development Board utilizing TrueSTUDIO

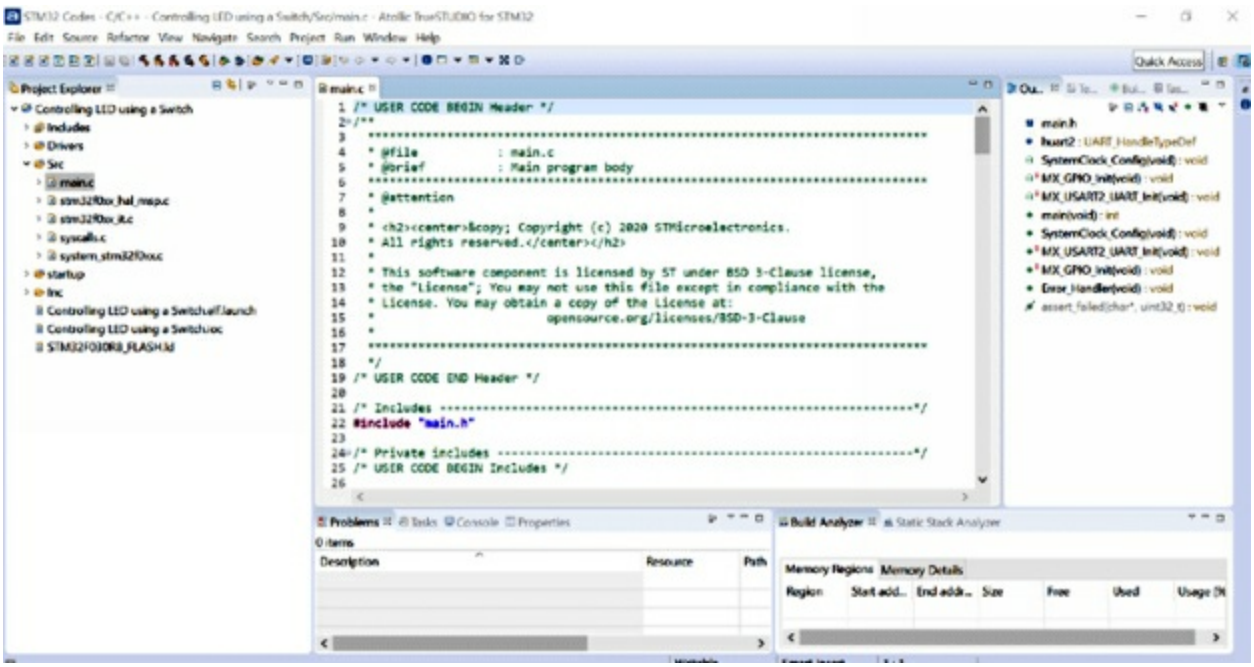
Presently your code or task will open in TrueSTUDIO consequently on the off chance that TrueSTUDIO requests workspace area, at that point furnish a workspace area or go with the default area.



The client will view the beneath given screen and afterward need to click at the corner mark in red shading.



What's more, presently we can see code in our TreuSTUDIO IDE. On the left side under the 'src' organizer we can see other program records (with .c expansion) that have just been produced for us from STM32Cube. We simply need to program the main.c document. Indeed, even in main.c document we will as of now have hardly any things set-up for us by the CubeMX we just need to alter it to suit our program. The total code inside the main.c record is given at the base of this page.



STM32 Nucleo64 Program to Control LED utilizing Push Button

Since all the necessary driver and code is created by STM32CubeMX, we just need to arrange a LED pin as yield and a press button as Input. The program for controlling drove utilizing the press catch ought to be written in the main.c record. The total program can be found at the base of this page. Its clarification is as per the following

We just have composed code for flipping the LED utilizing the press button. To accomplish this, we initially characterize pins for LED and press catches. Here we have distinguished a LED at Pin 5 number of PORTA

```
#define LED_PORT GPIOA

#define LED_PIN GPIO_PIN_5
```

What's more, distinguishe switch at Pin Number 13 of PORTC.

```
#define SW_PORT GPIOC
```

```
#define SW_PIN GPIO_PIN_13
```

At that point in the principle work, we have instated the every single utilized fringe.

```
/* Initialize all configured peripherals */

MX_GPIO_Init();

MX_USART2_Init();
```

And afterward read the press button utilizing the in the event that announcement and whenever discovered catch press (LOW) at that point LED will flip its state.

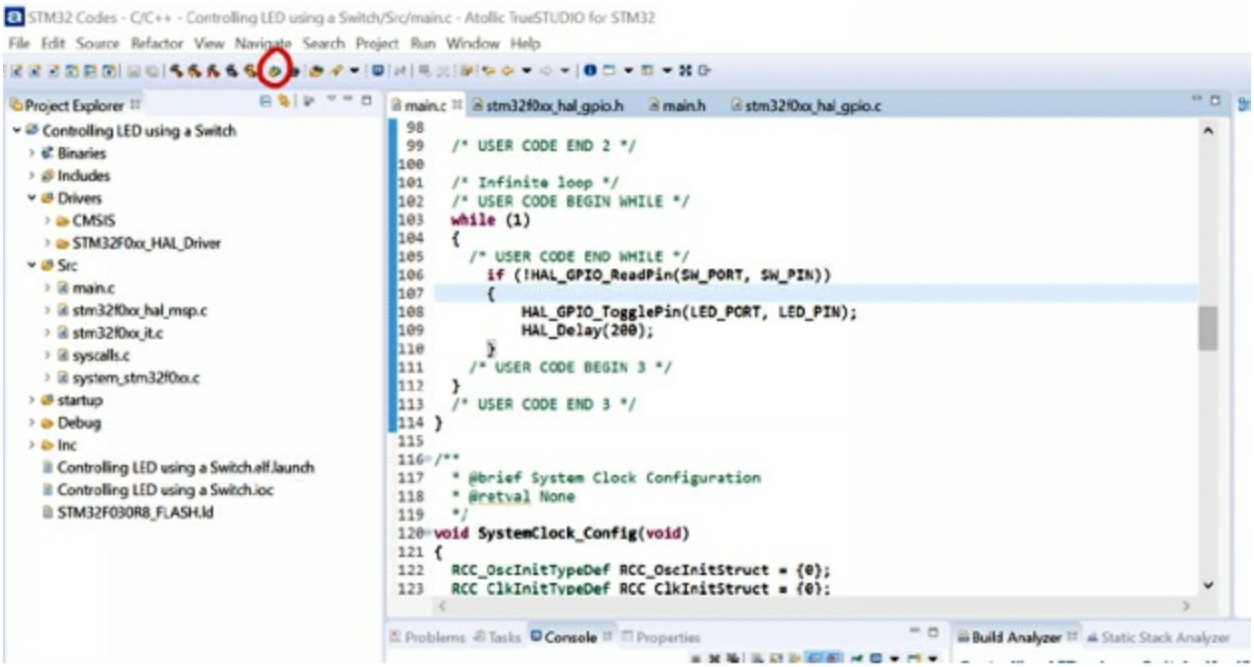
```
While (1)

{

/* USER CODE END WHILE */

If (!HAL_GPIO_ReadPin(SW_PORT, SW_PIN))

{

HAL_GPIO_TogglePin(SW_PORT, LED_PIN);

HAL_Delay(200);

}

/* USER CODE BEGIN 3 */

}
```

Here HAL_GPIO_ReadPin(SW_PORT, SW_PIN) work has two contentions, one is PORT and the other is a pin at which switch is associated and this pin is designed as INPUT while arranging fringe in STM32CubeMX.

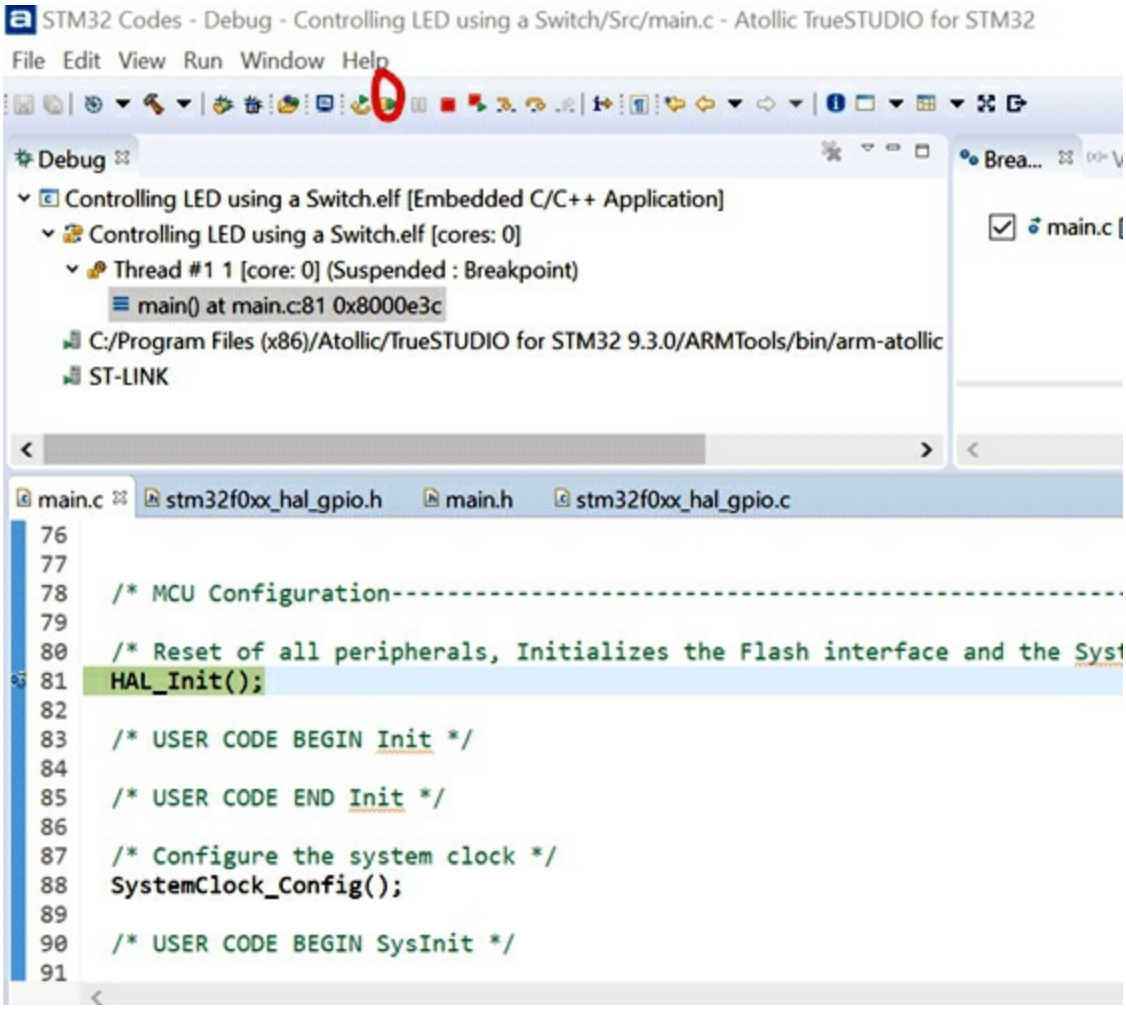
Investigating and Uploading Code to STM32 Necleo64 Development Board utilizing TrueSTUDIO

Presently associate your board to the PC utilizing the developer link. When you associate it, the driver required for the board should be naturally downloaded, you can check this using the gadget administrator.

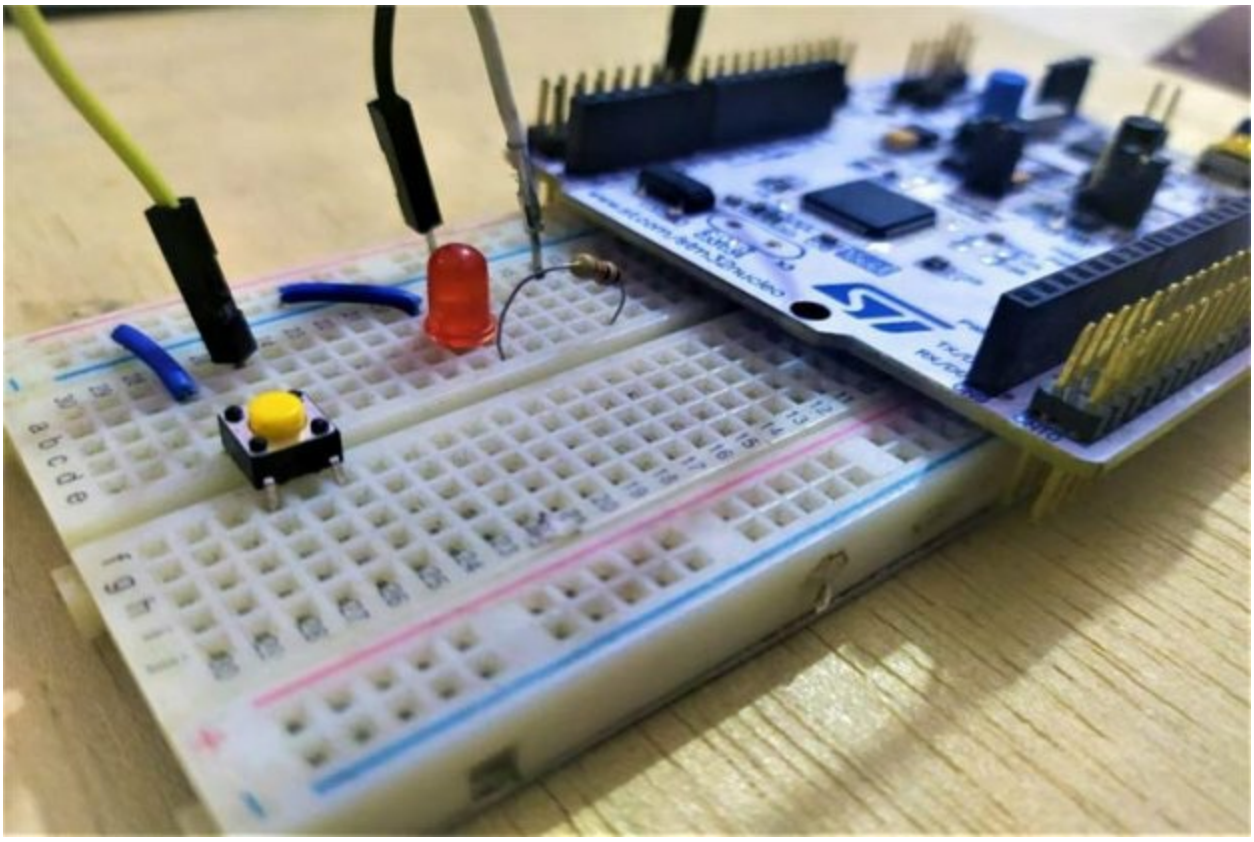
At that point, Press the investigate symbol set apart by the red hover in the underneath offered picture to gather the program and go into troubleshoot mode.



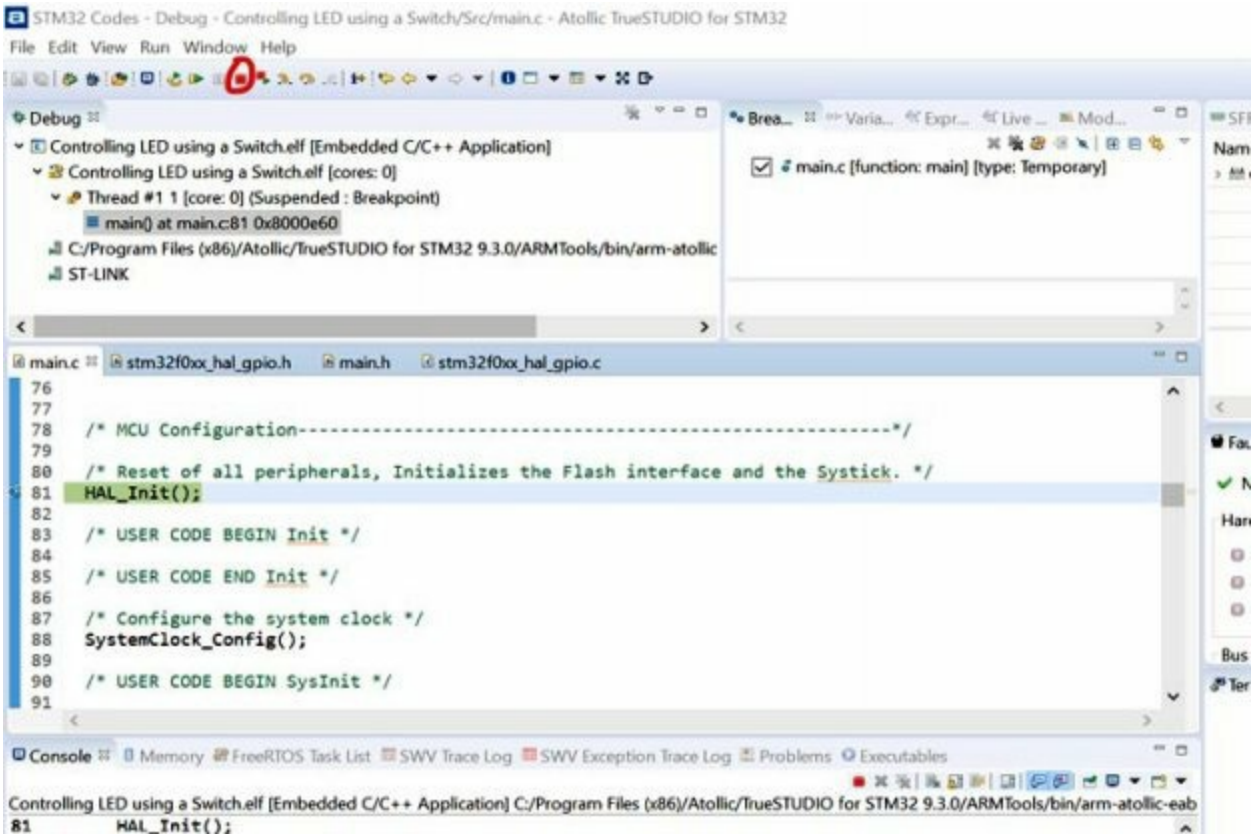
In investigate mode, the code will consequently be transferred. Presently we have to show the code to squeezing 'Resume' or F8 (set apart in the red circuit in the beneath picture).



Presently we can test the control of LED by squeezing the press button. As indicated by the code, the Light Emitting Diode should change its express every time you press the press button.



In the wake of testing, we can likewise end the program by squeezing the end symbol, set apart by the red hover in the beneath picture.



Code

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define LED_PORT GPIOA
#define LED_PIN GPIO_PIN_5
#define SW_PORT GPIOC
#define SW_PIN GPIO_PIN_13
/* USER CODE END PD */
```

```
/* Private macro -------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
  */
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();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
```

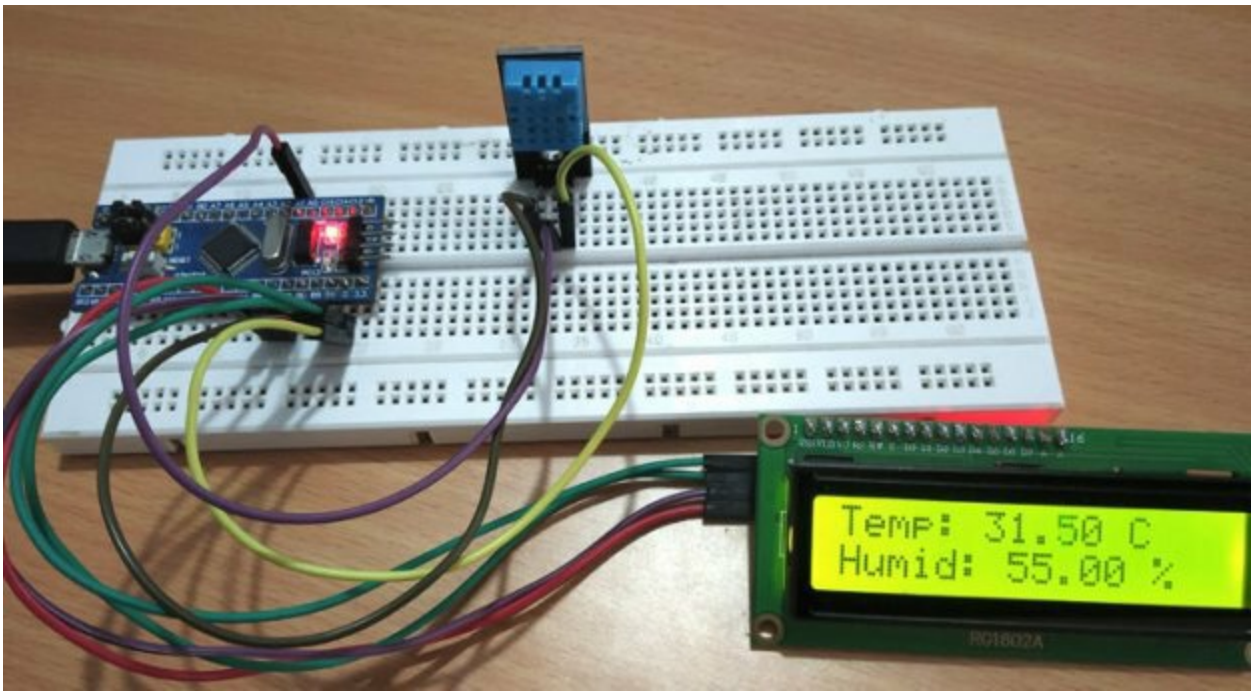
```
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
  if (!HAL_GPIO_ReadPin(SW_PORT, SW_PIN))
  {
  HAL_GPIO_TogglePin(LED_PORT, LED_PIN);
  HAL_Delay(200);
  }
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Initializes the CPU, AHB and APB busses clocks
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL12;
  RCC_OscInitStruct.PLL.PREDIV = RCC_PREDIV_DIV1;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
```

```
  /** Initializes the CPU, AHB and APB busses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1)
!= HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{
  /* USER CODE BEGIN USART2_Init 0 */
  /* USER CODE END USART2_Init 0 */
  /* USER CODE BEGIN USART2_Init 1 */
  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 38400;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
```

```
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */
  /* USER CODE END USART2_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
  /*Configure GPIO pin : PC13 */
  GPIO_InitStruct.Pin = GPIO_PIN_13;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
}
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
```

```
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state
*/
  /* 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(char *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****/
```

4. INTERFACING DHT11 TEMPERATURE ALONG WITH HUMIDITY SENSOR WITH STM32F103C8



DHT11 is a Temperature and moistness sensor which as the name infers is utilized to gauge the barometrical temperature along with mugginess in a specific situation or in a restricted shut space. The sensor is regularly utilized in observing natural parameters in numerous applications like Agriculture, Food Industries, Hospitals, Automobile, Weather Stations and so forth.

The sensor could quantify the temperature from 0°C to 50°C with a precision of 1°C. It is regularly utilized in controlled conditions, for example, Heat ventilation frameworks, temperature chambers and so on to screen temperature and take remedial measures. The estimating scope of mugginess is from 20% to 90% with an exactness of 1%. Stickiness shows the measure of water fume present noticeable all around. The estimation of moistness must be kept inside a controlled range in numerous situations like in assembling and putting away of tea powders the right stickiness must be kept up in the room or the tea will lose its taste and smell. The degree of stickiness in lounge rooms ought to likewise be kept up inside an agreeable range. The perfect estimation of dampness for most extreme solace is between half to 65%.

Today in this instructional exercise we will figure out how to interface the well known DHT11 Temperature along with Humidity sensor with STM32

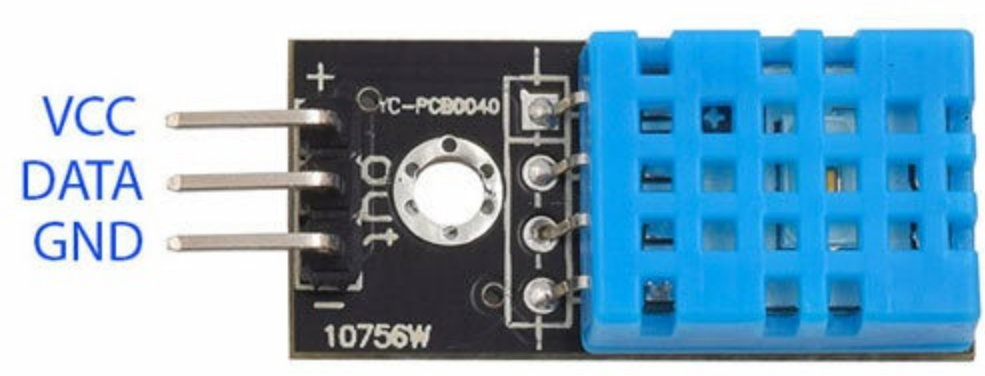
microcontroller. We have as of now took in the essentials of STM32 board and how to utilize it Arduino IDE. For the individuals who are new, the STM32 a.k.a BluePill advancement board, comprises of the STM32F103C8T6 microcontroller from ST Microelectronics. It is a 32-piece ARM Cortex M3 controller with high clock recurrence reasonable for fast and force limitation application. Verify all the STM32 related Tutorials here.

DHT11 Temperature along with Humidity Sensor

Before continuing with the interface system lets learn scarcely any things about the DHT11 sensor. As examined before, the DHT11 sensor is utilized to quantify Temperature and moistness. The sensor accompanies a committed in-constructed NTC to quantify temperature. It has a 8-piece microcontroller on-board to yield the estimations of temperature and stickiness as sequential information through one-wire convention. Which means, the sensor has just a single information pin through which both the temperature and mugginess esteems can be perused, consequently sparing pins on the microcontroller side. The sensor is additionally manufacturing plant adjusted and henceforth simple to interface with different microcontrollers.

DHT11 Specifications:

- Working Voltage: 3.5V to 5.5V
- Working current: 0.3mA (estimating) 60uA (reserve)
- Yield: Serial information
- Temperature Range: 0°C to 50°C
- Mugginess Range: 20% to 90%
- Goals: Temperature and Humidity both are 16-piece
- Precision: ±1°C and ±1%



No	Pin Name	Use
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	Ground	Connected to the ground of the circuit

We interfaced DHT11 sensor with numerous different microcontrollers to manufacture application like:

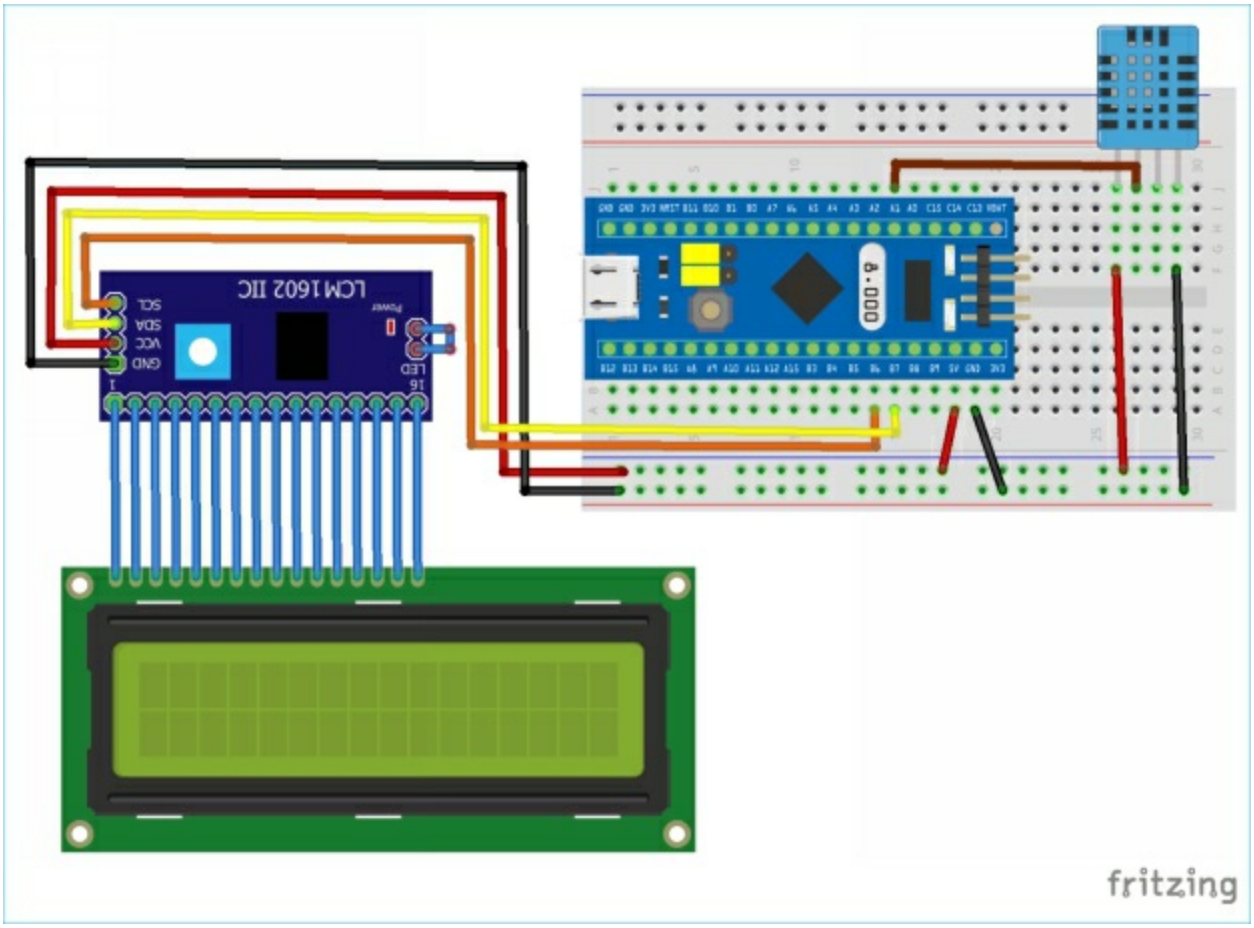
- Interfacing DHT11 Temperature along with Humidity Sensor with Raspberry Pi
- MATLAB Data Logging, Analysis along with Visualization: Plotting DHT11 Sensor readings on MATLAB
- Programmed AC Temperature Controller utilizing Arduino, DHT11 along with IR Blaster
- IoT Weather Station utilizing NodeMCU: Monitoring Humidity, Temperature along with Pressure over Internet

Segments Required

- STM32F103C8
- DHT11 Temperature along with Humidity Sensor
- 16x2 Liquid Crystal Display Display
- IIC/I2C Serial Interface Adapter Module
- Breadboard
- Associating Wires

Schematic Diagram

The total circuit graph to interface DHT11 with STM32 Microcontroller is demonstrated as follows. The circuit was drawn utilizing Fritzing programming.



As should be obvious, we have utilized an I2C interface module to associate the LCD module to STM32. This makes the associations straightforward and further diminishes the quantity of pins utilized on the controller side. Nonetheless in case you don't have this module you can similarly interface LCD legitimately STM32 by following the connection.

In the event that you have an interface module, at that point the circuit Connections between I2C Serial Interface Module (Fixed with 16X2 LCD Display) and STM32F103C8 is classified underneath:

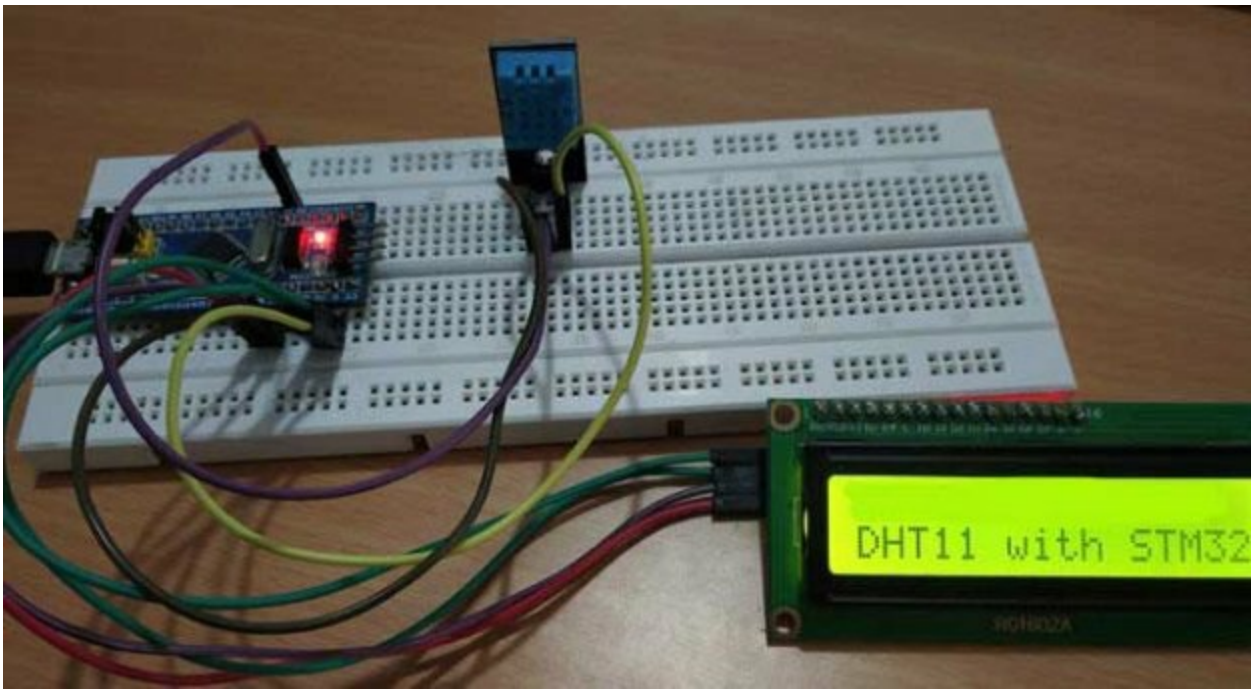
I2C Serial Interface Module	**STM32F103C8**
VCC	5V

GND	GND
SDA	PB7
SCL	PB6

Also the circuit Connections somewhere in the range of STM32F103C8 and DHT11 Sensor is arranged beneath.

DHT11	**STM32F103C8**
VCC	5V
GND	GND
Data	PA1

When the associations are done my equipment looks something like this beneath.



The whole set-up is controlled by the USB port of STM32 from my PC. Since our equipment is prepared, lets get into the coding part.

Setting up the Arduino IDE for STM32F103C8

We need to compose a program to peruse the temperature and dampness esteem from the DHT11 senor and show it on the LCD module. Here the LCD show is associated through I2C connector, thus we have first discover the I2C address of this connector to speak with LCD.

Interfacing I2C Serial LCD Interface Adapter Module with STM32F103C8:

From circuit outline we can view that the STM32F103C8 I2C pins PB6 and PB7 are associated with the SCL along with SDA pins of I2C Serial Interface Module. To discover the location of the I2C Serial Interface module we need to check for accessible locations.

Checking the Address of the I2C Serial Interface Module:

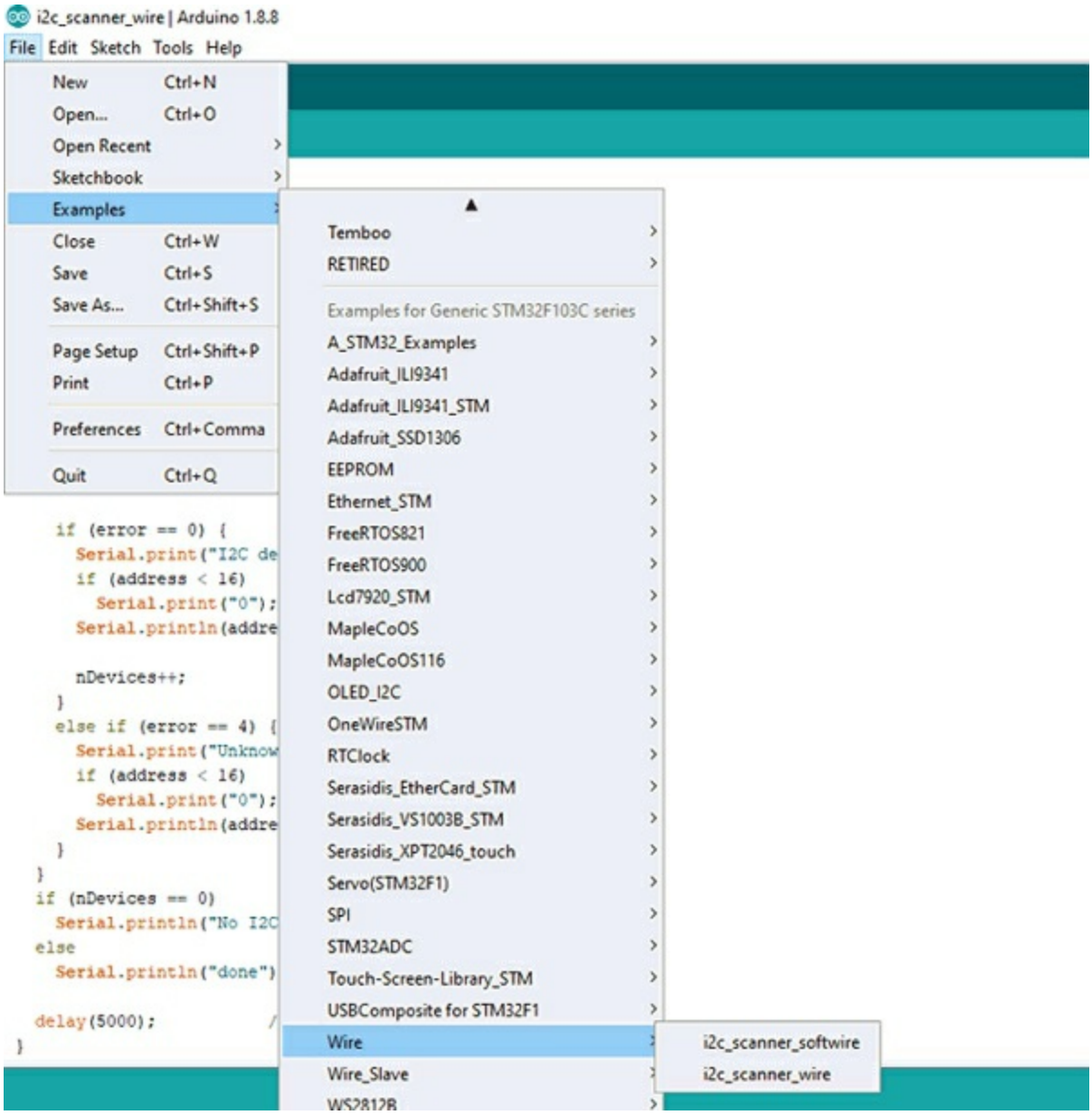
Follow the beneath steps to discover the I2C address of your LCD I2C

interface module.

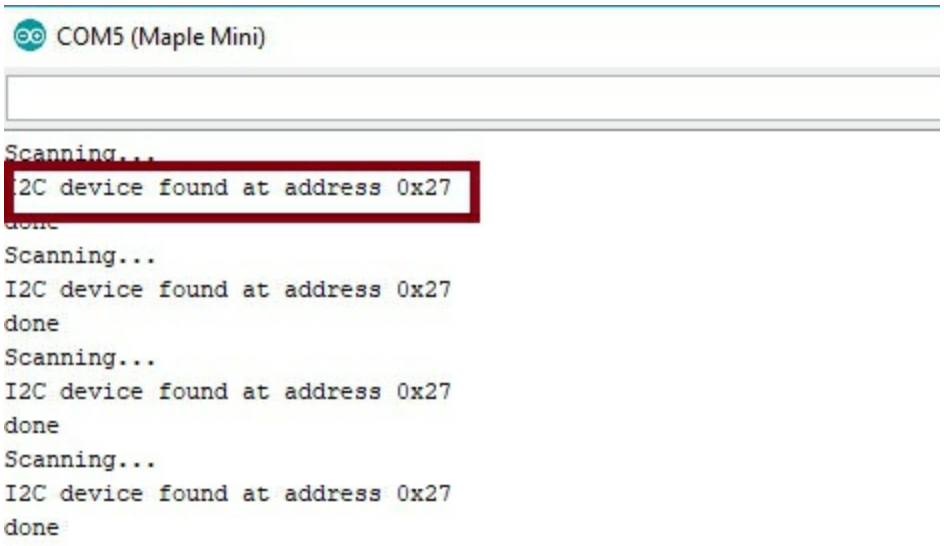
1. First check the STM32 bundle for Arduino IDE is introduced. If not follow the connection Programming your STM32 in ARDUINO IDE.

2. While introducing bundles for programming STM32 utilizing Arduino IDE by above connection the wire library is introduced in default.

3.Program for checking the I2C gadget associated is available in the models (In Arduino IDE: Files->Examples->Wire->I2C scanner wire). Before that choose the board in Tools->Board->Generic STM32F103C8 Series as demonstrated as follows.



4. After then transfer the code to the STM32F103C8 and the open sequential screen.



Presently note the I2C address of the I2C 16x2 LCD show as (0x27).

Introducing Library for I2C 16x2 Display Module along with DHT11 Sensor:

Since we realize the I2C address we have to installed a library for conveying to the LCD show across I2C. The I2C Liquid Crystal Display show library can be installed from this connection. In the wake of downloading the compress record introduce I2C LCD library in the Arduino IDE by sketch->import library.This library can likewise be utilized with Arduino sheets for speaking with I2C LCD show modules.

Likewise so as to peruse the sequential information from DHT11 sensor we will utilize the DHT11 library. Download the library as a ZIP record utilizing the connection gave and subsequent to downloading, introduce DHT library in the Arduino IDE by utilizing sketch->import library. Again a same library can likewise be utilized with Arduino sheets.

Coding Explanation

The total code for this article can be found at the base of this page, the clarification for the equivalent is as per the following. At first incorporate the necessary libraries. Incorporate Wire.h library for utilizing I2C in STM32F103C8, LiquidCrystal_I2C.h for utilizing I2C type LCD show and DHT.h for utilizing DHT sensor capacities

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>

#include <DHT.h>
```

Presently the pin name of DHT11 (OUT pin) that is associated with the PA1 of STM32F103C8 is characterized

```
#define DHTPIN PA1
```

And furthermore, the DHTTYPE is characterized as DHT11.

```
#define DHTTYPE DHT11
```

Next the article lcd for class LiquidCrystal_I2C with I2C address of 0x27 and 16x2 sort LCD show is introduced.

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

And furthermore, object dht for class DHT with DHT pin with STM32 and DHT type as DHT11 is instated

```
DHT dht(DHTPIN, DHTTYPE);
```

Next in void arrangement ():

First instate the LCD utilizing following

```
lcd.begin();
```

And afterward use underneath proclamation to start to get Temperature and

mugginess esteems from DHT11 sensor.

```
dht.begin();
```

Turn on the blacklight and print an invite message and clear them following three seconds.

```
lcd.backlight();

lcd.setCursor(0,0);

lcd.print("Hello_world");

lcd.setCursor(0,1);

lcd.print("DHT11 with STM32");

delay(3000);

lcd.clear();
```

Next in the void circle():

The worth is gotten from the DHT11 sensor constantly. So as to get the different estimations of temperature along with moistness and store it in a variable after articulation is utilized.

To get just the Humidity esteem

```
float h = dht.readHumidity();
```

To get just the Temperature esteem

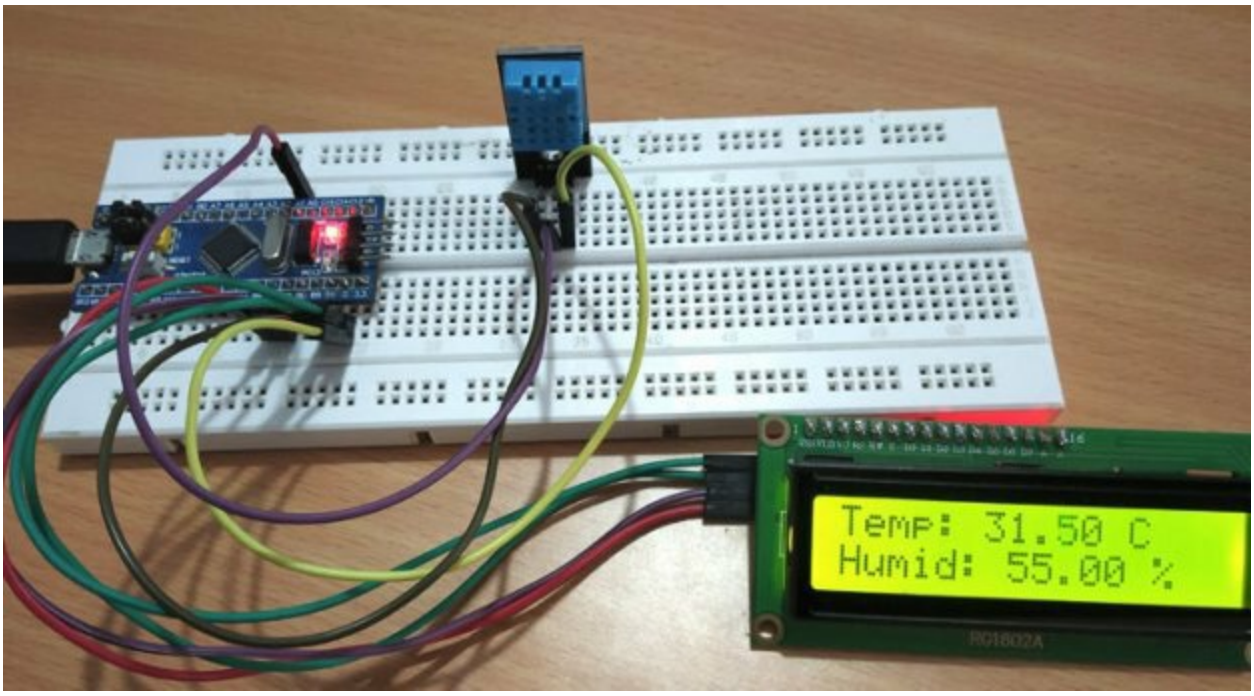
```
float t = dht.readTemperature();
```

Lastly print those in the 16X2 LCD show

```
lcd.setCursor(0,0);

lcd.print("Temp: ");

lcd.print(t);

lcd.print(" C");

lcd.setCursor(0,1);

lcd.print("Humid: ");

lcd.print(h);

lcd.print(" %");
```

Working of STM32 based Thermometer

When your equipment and code is prepared, only the code to your equipment and you should see your LCD showing the welcome screen followed by the constant Temperature and Humidity esteems as demonstrated as follows



In the event that your showcases shows nothing you can check altering the differentiation potentiometer at the rear of the I2C module. I had a go at different my room temperature utilizing an Air conditioner and found the sensor incentive to likewise change in like manner. The AC additionally has a choice to gauge room temperature and as should be obvious in underneath picture my remote shows the room temperature to be 27°C and our sensors likewise shows 27.3°C on the LCD which is practically close.

Expectation you comprehended the task and appreciated structure it.

Code

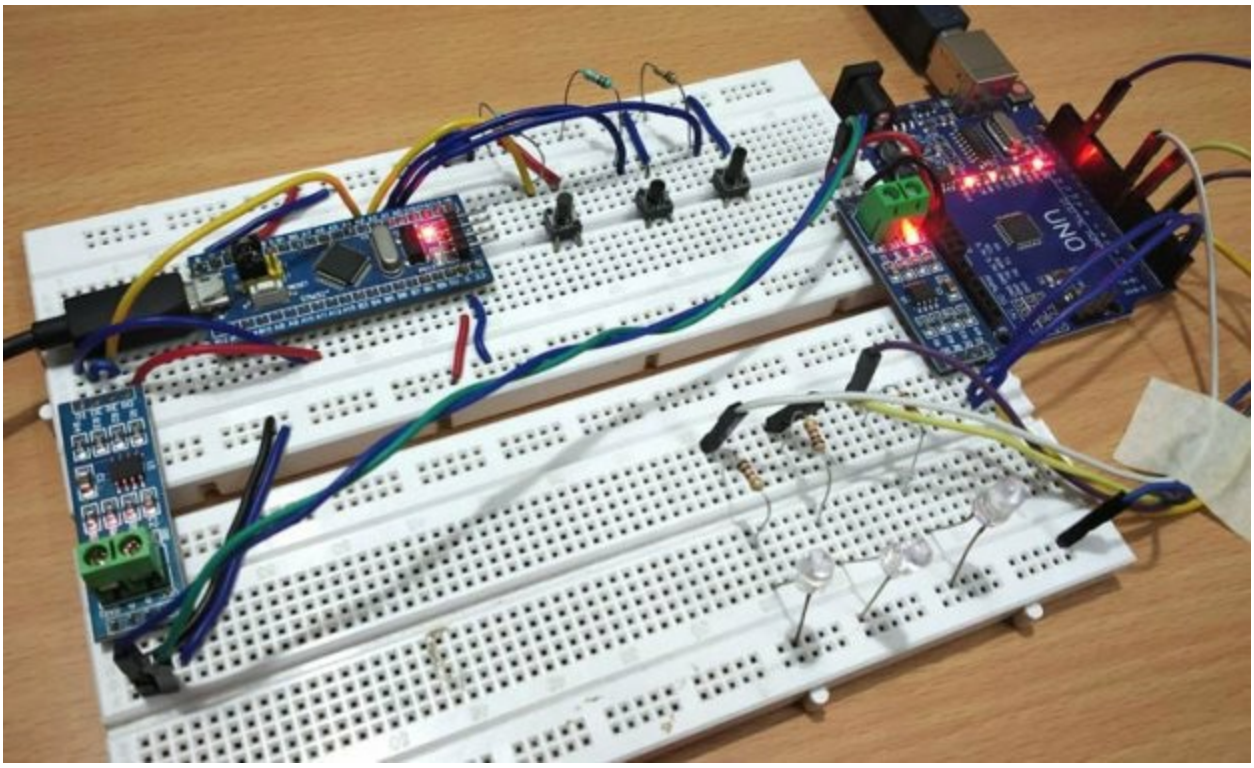
```
#include <Wire.h>      //Library for using I2C
#include <LiquidCrystal_I2C.h>  //Library for using I2C type LCD display
#include <DHT.h>                  //Library for using DHT sensor

#define DHTPIN PA1

#define DHTTYPE DHT11

LiquidCrystal_I2C lcd(0x27, 16, 2);  //initilize object lcd for class LiquidCrystal_I2C with I2C address of 0x27 and 16x2 type LCD display

DHT dht(DHTPIN, DHTTYPE);     //initilize object dht for class DHT with DHT pin with STM32 and DHT type as DHT11

void setup()
{
  // initialize the LCD
```

```
  lcd.begin();
  dht.begin();          //Begins to receive Temperature and humidity
values.
  lcd.backlight();      // Turn on the blacklight and print a welcome message.
  lcd.setCursor(0,0);
  lcd.print("Hello_world");
  lcd.setCursor(0,1);
  lcd.print("DHT11 with STM32");
  delay(3000);
  lcd.clear();
}
```

```
void loop()
{
  float h = dht.readHumidity();       //Gets Humidity value
  float t = dht.readTemperature();    //Gets Temperature value
  lcd.setCursor(0,0);
  lcd.print("Temp: ");
  lcd.print(t);
  lcd.print(" C");
  lcd.setCursor(0,1);
  lcd.print("Humid: ");
  lcd.print(h);
  lcd.print(" %");
}
```

5. SEQUENTIAL COMMUNICATION BETWEEN STM32F103C8

AND ARDUINO UNO UTILIZING RS-485



The correspondence conventions are the indispensable piece of a computerized hardware and implanted framework. Any place there is interfacing of different microcontroller and peripherals, the correspondence convention must be utilized so as to trade pack of information. There are numerous sorts of sequential correspondence convention accessible. The RS485 is the sequential correspondence convention and is utilized in mechanical tasks and substantial hardware.

We found out about RS485 Serial Communication among Arduino Uno along with Arduino Nano in the past instructional exercise. This instructional exercise is tied in with utilizing a RS-485 Serial correspondence in STM32F103C8 Microcontroller. In the event that you are new to STM32 Microcontroller, at that point start with Getting Initiated with STM32 utilizing Arduino IDE: Blinking LED and check all the STM32 extends here.

In this instructional exercise Master STM32F103C8 has three press fastens that are utilized to control the status of three LEDs present at the Slave Arduino Uno by utilizing RS-485 Serial correspondence.

How about we start by understanding the working of RS-485 Serial

correspondence.

RS-485 Serial Communication

RS-485 is a nonconcurrent sequential correspondence convention which doesn't require clock. It utilizes a system called differential sign to move paired information starting with one gadget then onto the next.

So what is this Differential Signal Transfer Method??

Differential sign technique works by making a differential voltage by utilizing a positive and negative 5V. It gives a Half-Duplex correspondence when using 2 wires and Full-Duplex correspondence when utilizing four wires.

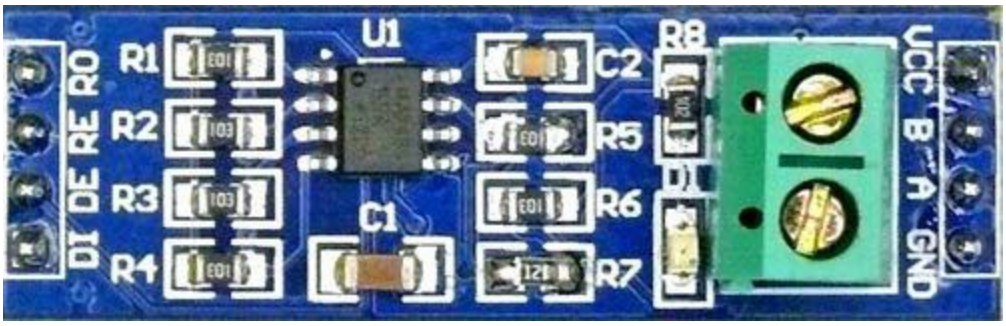
By utilizing this strategy:

- RS-485 backings higher information move pace of 30Mbps most extreme.
- It additionally gives most extreme information move separation contrasted with RS-232 convention. It moves information up to 1200-meter most extreme.
- The primary preferred position of RS-485 over RS-232 is the numerous slave with single Master while RS-232 backings just single slave.
- Can have a limit of 32 gadgets associated with RS-485 convention.
- Another bit of leeway of the RS-485 is resistant to the commotion as they utilize differential sign strategy to move.
- RS-485 is quicker contrasted with I2C convention.

RS-485 Module can be associated with any microcontroller having sequential port. For utilizing RS-485 module with microcontrollers a module called 5V MAX485 TTL to RS485 which depends on Maxim MAX485 IC is required

as it permits sequential correspondence over significant distance of 1200 meters and it is bidirectional and half duplex has an information move pace of 2.5 Mbps. This module requires a voltage of 5V.

RS-485 Pin Description:



Pin Name	Description
VCC	5V
A	Non-inverting Receiver Input Non-Inverting Driver Output
B	Inverting Receiver Input Inverting Driver Output
GND	GND (0V)
R0	Receiver Out (RX pin)
RE	Receiver Output (LOW-Enable)
DE	Driver Output (HIGH-Enable)
DI	Driver Input (TX pin)

RS485 module has following highlights:

- Working voltage: 5V
- On-board MAX485 chip
- A low force utilization for the RS485 correspondence
- Slew-rate restricted handset
- 5.08mm pitch 2P terminal
- Advantageous RS-485 correspondence wiring
- All pins of chip have been lead to can be controlled through the microcontroller
- Board size: 44 x 14mm

Utilizing this module with STM32F103C8 and Arduino UNO is exceptionally simple. The equipment sequential ports of microcontrollers are utilized. The equipment sequential pins in STM32 and arduino UNO is given beneath.

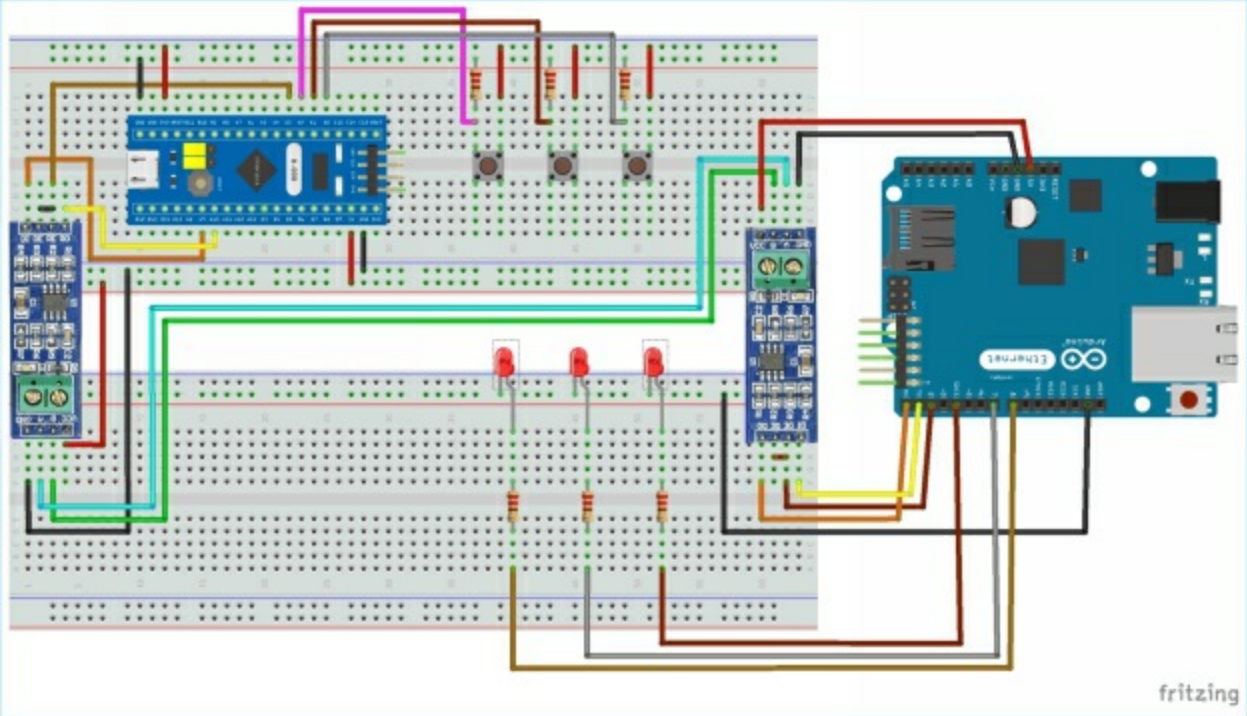
- In STM32F103C8: Pins PA9 (TX) along with PA10 (RX)
- In Arduino Uno: Pin 0 (RX) and 1 (TX)

Writing computer programs is additionally basic simply utilize the Serial.print() to keep in touch with RS-485 and Serial.Read() to peruse from RS-485 and the pins DE and RE of RS-485 is made LOW to get information and made HIGH to compose information to RS-485 transport.

Segments Required

- STM32F103C8
- Arduino UNO
- MAX485 TTL to RS485 Converter Module - (2)
- 10K Potentiometer
- Press Button - 3
- Driven - 3
- Resistors
- Breadboard
- Associating Wires

Circuit Diagram



In this instructional exercise STM32F103C8 is utilized as Master with one RS-485 module and Arduino UNO is utilized as Slave with another RS-485 module.

Circuit Connection between the RS-485 and STM32F103C8 (Master):

RS-485	**STM32F103C8**
DI	PA9 (TX1)
DE RE	PA3
R0	PA10 (RX1)
VCC	5V
GND	GND
A	To A of Slave RS-485
B	To B of Slave RS-485

STM32F103C8 with Three Push button:

Three Push catches with three Pull Down Resistor of 10k are associated with the pins PA0, PA1, PA2 of STM32F103C8.

Circuit Connection between the RS-485 and Arduino UNO (Slave):

RS-485	Arduino UNO
DI	1 (TX)
DE RE	2
R0	0 (RX)
VCC	5V
GND	GND
A	To A of Master RS-485
B	To B of Master RS-485

Three LEDs are utilized where Anodes of LEDs with resistor of 330 ohms are associated with pins 4, 7, 8 of Arduino UNO along with Cathode of the Light Emitting Diodes are associated with GND.

Programming STM32F103C8 and Arduino UNO for RS485 Serial Communication

Arduino IDE is utilized for improvement and programming of the two sheets for example STM32 and Arduino UNO. In any case, ensure you have chosen the comparing PORT from Tools->Port and Board from Tools->Board. In the event that you discover any challenges or uncertainty, at that point simply allude Programming your STM32 in ARDUINO IDE. The programming for this instructional exercise comprises of Two segment one for STM32F103C8 (Master) and other for Arduino UNO (Slave). Both the codes will be clarified

individually beneath.

STM32F103C8 as Master

In Master side, the status of the Push Button is perused and afterward sequentially composed those qualities to the RS-485 transport through the Hardware Serial Ports 1 (PA9, PA10) of STM32F103C8. Additionally there is no outside library required starting at now. The Arduino has all the fundamental library for sequential correspondence.

Start Serial Communication utilizing Hardware Serial Pins (PA9, PA10) at buadrate of 9600.

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

Peruse the status of the press button at the pins PA0, PA1, PA2 of STM32F103C8 and store them in a variable button1val, button2val, button3val. The worth is HIGH if button is compressed and LOW when not compressed.

```
int button1val = digitalRead(button1);

int button2val = digitalRead(button2);

int button3val = digitalRead(button3);
```

Before sending the catch esteems to the sequential port, the pins DE and RE of RS-485 ought to be HIGH that is associated with the pin PA3 of STM32F103C8 (To Make pin PA3 HIGH):

```
digitalWrite(enablePin, HIGH);
```

Close to place those qualities in the Serial Port and send values relying on which press button is squeezed use if else proclamation and send the

comparing esteem when catch is compressed.

In the event that the principal button is squeezed, at that point the condition matches and the worth '1' is sent to the sequential port where Arduino UNO is associated.

```
if (button1val == HIGH)
    {
      int num1 = 1;
      Serial1.println(num1);
    }
```

Additionally, when button 2 is squeezed the worth 2 is sent over sequential port and when button 3 is squeezed the worth 3 is sent over the sequential port.

```
else if (button2val == HIGH)
    {
      int num2 =2;
      Serial1.println(num2);
    }
                else if (button3val == HIGH)
    {
      int num3 =3;
```

```
    Serial1.println(num3);
  }
```

What's more, when no catch is compressed the worth 0 is sent to Arduino Uno.

```
else
  {
    int num = 0;
    Serial1.println(num);
    }
```

This completes the process of programming to arrange STM32 as Master.

Arduino UNO as Slave

In the Slave side the Arduino UNO gets a whole number worth that is sent from the Master STM32F103C8 which is accessible at the Hardware Serial port of the Arduino UNO (P0, 1) where the RS-485 module is associated.

Just read worth and store in a variable. Contingent on the worth got the relating LED is turned ON otherwise OFF associated with Arduino GPIO.

To get the qualities from the Master simply make the pins DE and RE of RS-485 module LOW. So the pin-2 (enablePin) of Arduino UNO is made LOW.

```
digitalWrite(enablePin, LOW);
```

Presently simply read the whole number information accessible at Serial Port and store them in a variable.

```
int receive = Serial.parseInt();
```

Contingent on the worth for example (0, 1, 2, 3) got, the correspondingly one of the three LED is turned ON.

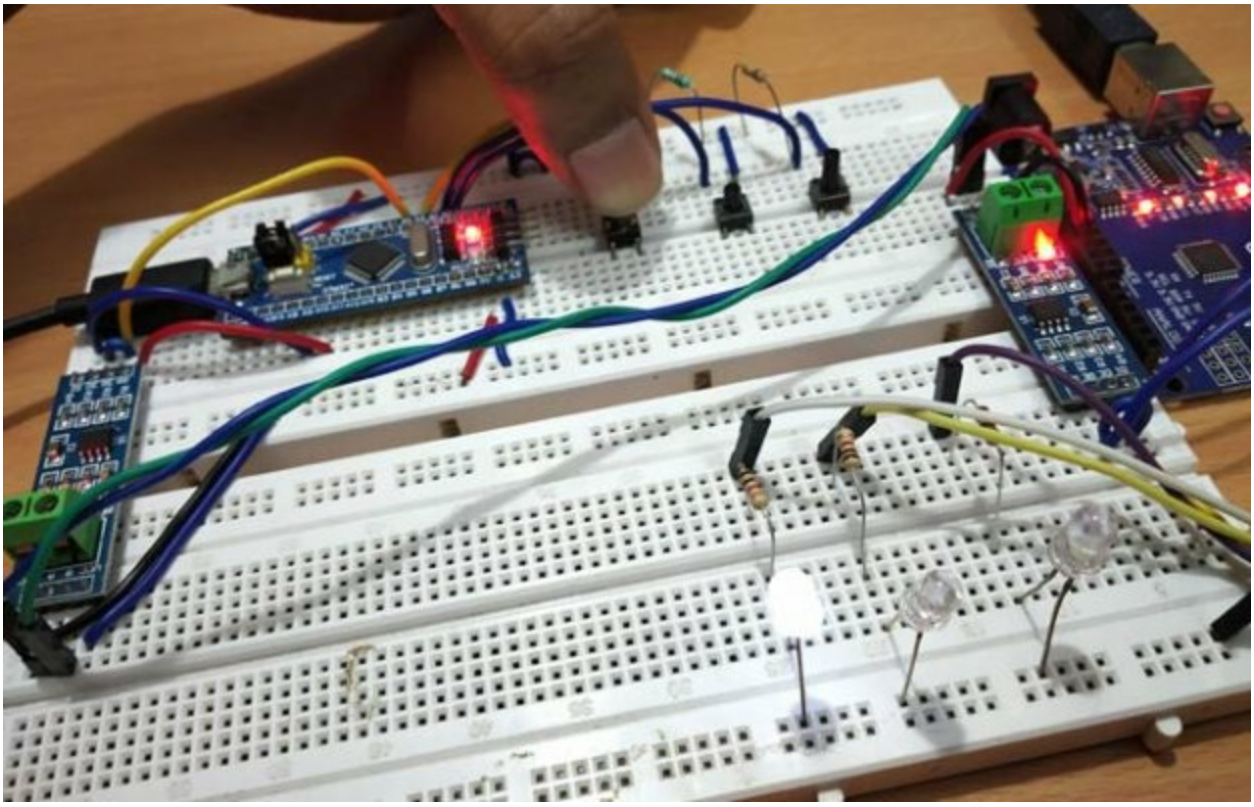
```
if (receive == 1)                    //Depending Upon Recieved value the corresponding LED is turned ON or OFF

        {

          digitalWrite(ledpin1,HIGH);

        }

        else if (receive == 2)

        {

          digitalWrite(ledpin2,HIGH);

        }

        else if (receive == 3)

        {

          digitalWrite(ledpin3,HIGH);

        }

        else

        {
```

```
        digitalWrite(ledpin1,LOW);

        digitalWrite(ledpin2,LOW);

        digitalWrite(ledpin3,LOW);

      }
```

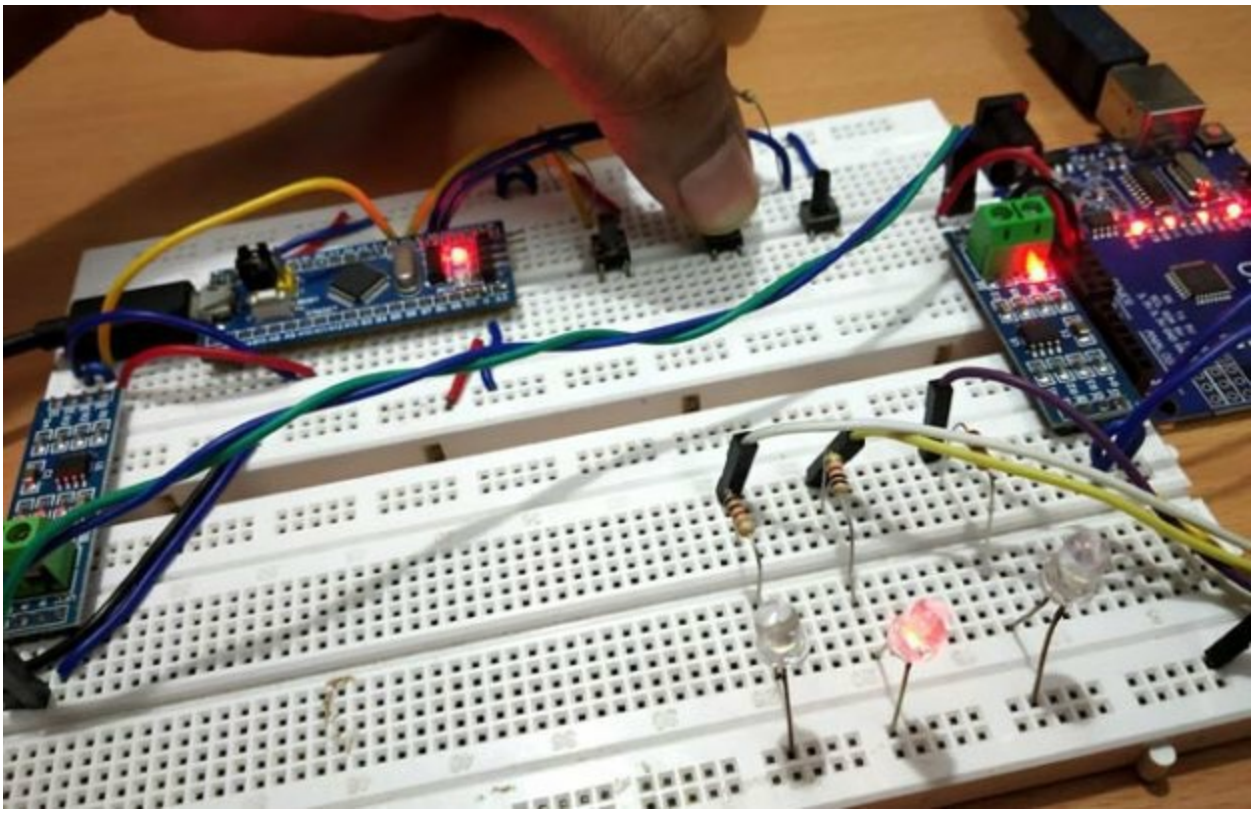
This completes the process of programming and arranging Arduino UNO as Slave. Additionally this completes the total designs for Arduino UNO and STM32. All codes are appended over the end of this instructional exercise.

Testing the RS485 correspondence among STM32F103C8 and Arduino UNO:

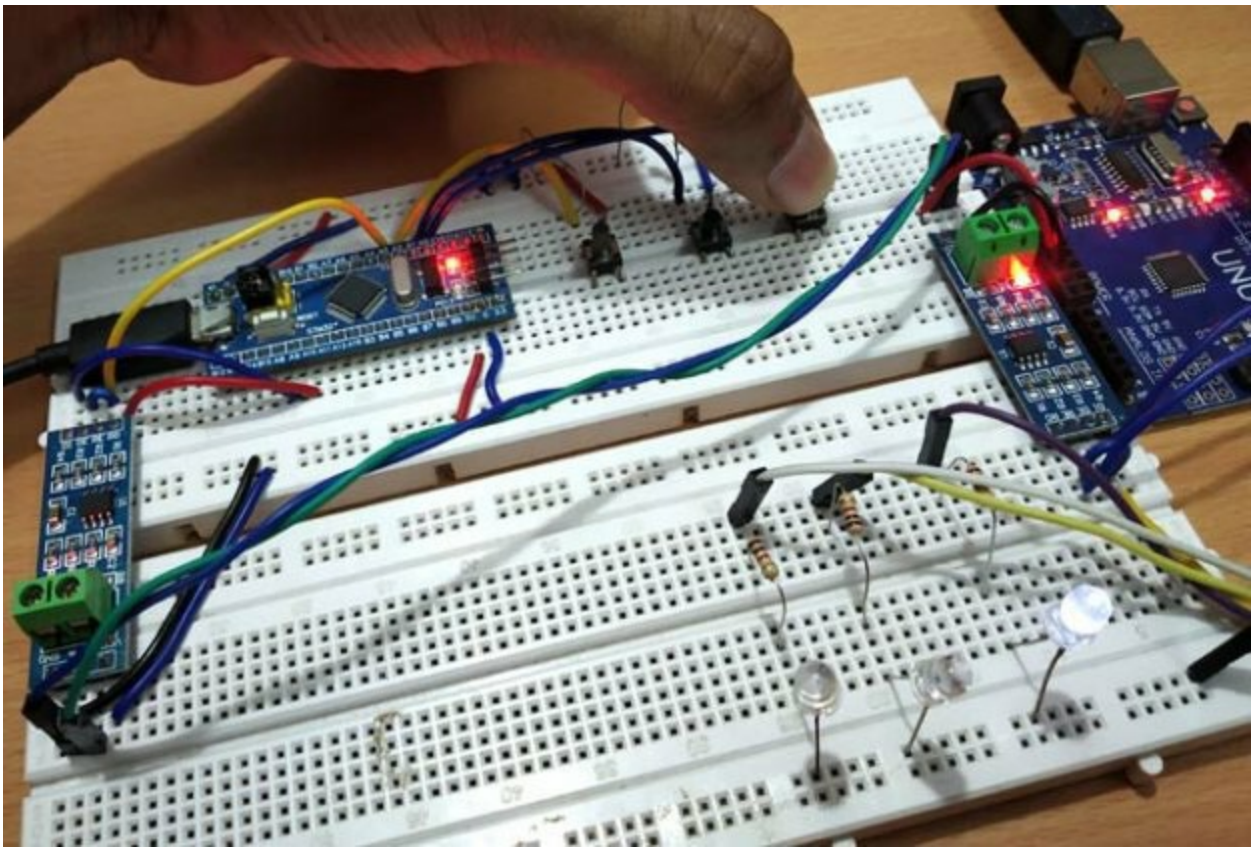
1. At the point when Push button-1, which is associated with the Master STM32, is squeezed the LED 1 Turns ON associated with the Slave Arduino.



2. At the point when Push button-2, which associated with the Master STM32, is squeezed the LED 2 Turns ON associated with the Slave Arduino.



3. Correspondingly when Push button-3 is squeezed the LED 3 Turns ON associated with the Slave Arduino.



This completes the RS485 sequential correspondence among STM32F103C8 and Arduino UNO. The Arduino UNO and STM32 sheets are generally utilized sheets for quick prototyping and we have done numerous valuable activities on these sheets.

Code

Master Code:STM32F103C8

```
//RS-485 Serial Communication Between STM32F103C8 & Arduino Uno

#define button1 PA0
#define button2 PA1
#define button3 PA2
#define enablePin PA3


void setup()
```

```
{
  Serial1.begin(9600);            // Begins Serial communication at serial1 Port PA9,PA10 at baudrate 9600
  pinMode(enablePin, OUTPUT);
```

```
  pinMode(button1,INPUT);
  pinMode(button2,INPUT);
  pinMode(button3,INPUT);


   delay(10);
  digitalWrite(enablePin, HIGH);  //  (always high as Master Writes data to Slave)
}
```

```
void loop()
{
  int button1val = digitalRead(button1);         //Reads the status of the Push Button
  int button2val = digitalRead(button2);
  int button3val = digitalRead(button3);


     if (button1val == HIGH)
    {
      int num1 = 1;
      Serial1.println(num1);               //Sends Push button value 1 if HIGH (Pressed)
    }
    else if (button2val == HIGH)
    {
      int num2 =2;
      Serial1.println(num2);               //Sends Push button value 2 if HIGH (Pressed)
    }
    else if (button3val == HIGH)
    {
      int num3 =3;
      Serial1.println(num3);               //Sends Push button value 3 if HIGH
```

```
(Pressed)
    }
   else
   {
     int num = 0;
     Serial1.println(num);                //Sends 0 if any of push button is not
pressed
   }
   delay(50);
}
```

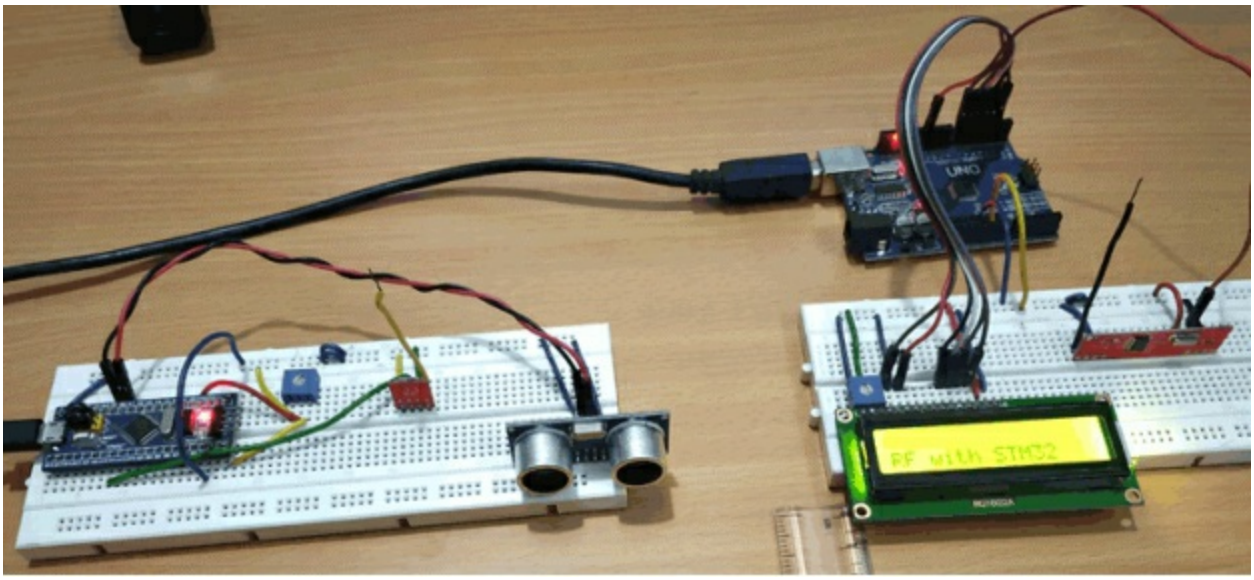
Slave Code: Arduino UNO:

```
//RS-485 Serial Communication Between STM32F103C8 & Arduino Uno

#define enablePin 2
#define ledpin1 4
#define ledpin2 7
#define ledpin3 8

 void setup()
{
  Serial.begin(9600);                 // Begins Serial Communication with baud
rate 9600
  pinMode(ledpin1,OUTPUT);
  pinMode(ledpin2,OUTPUT);
  pinMode(ledpin3,OUTPUT);
  pinMode(enablePin, OUTPUT);
  delay(10);
  digitalWrite(enablePin, LOW);        //  (Pin 8 always LOW to receive value
from Master)
}

void loop()

{
  while (Serial.available())           //While having data at Serial port this loop
executes
     {
       int receive = Serial.parseInt();  // Reads the integer value sent from
STM32
```

```
          if (receive == 1)              //Depending Upon Recieved value the
corresponding LED is turned ON or OFF
        {
          digitalWrite(ledpin1,HIGH);
        }
        else if (receive == 2)
        {
          digitalWrite(ledpin2,HIGH);
        }
        else if (receive == 3)
        {
          digitalWrite(ledpin3,HIGH);
        }
        else
        {
         digitalWrite(ledpin1,LOW);
         digitalWrite(ledpin2,LOW);
         digitalWrite(ledpin3,LOW);
        }
    }
}
```

◆ ◆ ◆

6. INTERFACING 433MHZ RF MODULE WITH STM32F103C8



Making remote undertakings in implanted hardware turns out to be significant and accommodating as there are no disordered wires all over which makes the gadget progressively helpful and convenient. There are different remote innovations, for example, Bluetooth, WiFi, 433 MHz RF (Radio Frequency) and so on. Each innovation has its own focal points and weaknesses, for example, cost, separation or range move, speed or throughput and so on. Today we will utilize RF module with STM32 to send along with get the information remotely. On the off chance that you are new to STM32 Microcontroller, at that point start with Blinking LED with STM32 utilizing Arduino IDE and check all other STM32 extends here.

Aside from this, we have additionally utilized RF 433Mhz Wireless Module with different microcontrollers to assemble some remote controlled tasks, for example,

- RF Controlled Home Appliances
- RF Remote Controlled Light Emitting Diodes Using Raspberry Pi
- RF Controlled Robot
- Interfacing RF module with Arduino

- PIC to PIC Communication utilizing RF Module

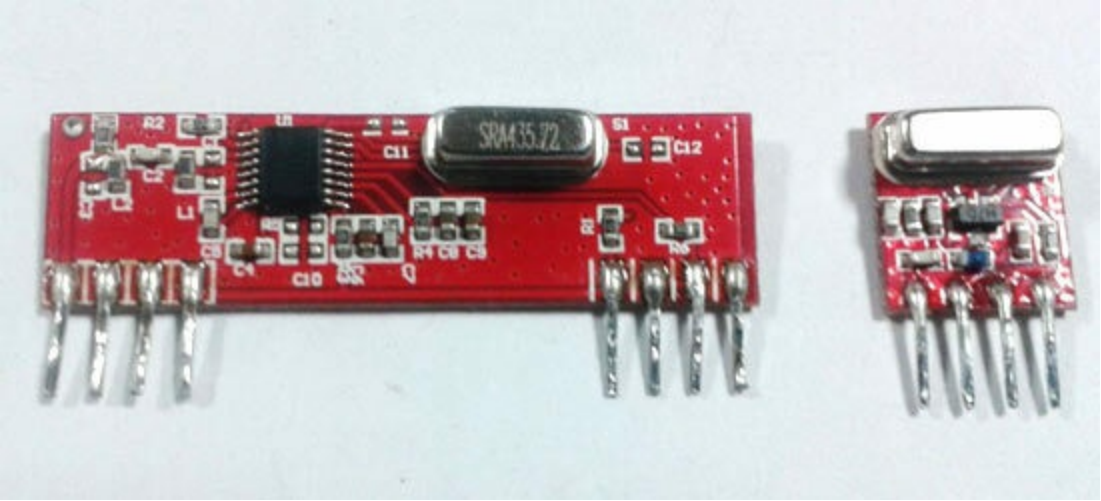
Here we will interface a 433MHz RF remote module with STM32F103C8 microcontroller. The undertaking is isolated into two sections. The transmitter will be interfaced with STM32 along with the collector will be interfaced with Arduino UNO. There will be distinctive circuit chart and outlines for both transmitting also getting part.

In this instructional exercise, RF Transmitter sends two qualities to Receiver side: the separation estimated utilizing ultrasonic sensor and the potentiometer ADC esteem (0 to 4096) which is mapped as number from (0 to 100). The RF collector of Arduino gets both the qualities and prints those separation and number qualities in 16x2 LCD show remotely.

Parts Required

- STM32F103C8 Microcontroller
- Arduino UNO
- 433Mhz RF Transmitter along with Receiver
- Ultrasonic Sensor (HC-SR04)
- 16x2 LCD show
- 10k Potentiometer
- Breadboard
- Interfacing Wires

433Mhz RF Transmitter along with Receiver Module)



RF Transmitter Pinout:

433Mhz RF Transmitter	**Pin Description**
ANT	For connecting Antenna
GND	GND
VDD	3.3 to 5V
DATA	Data to be transmitted to receiver is given here

RF Receiver Pinout:

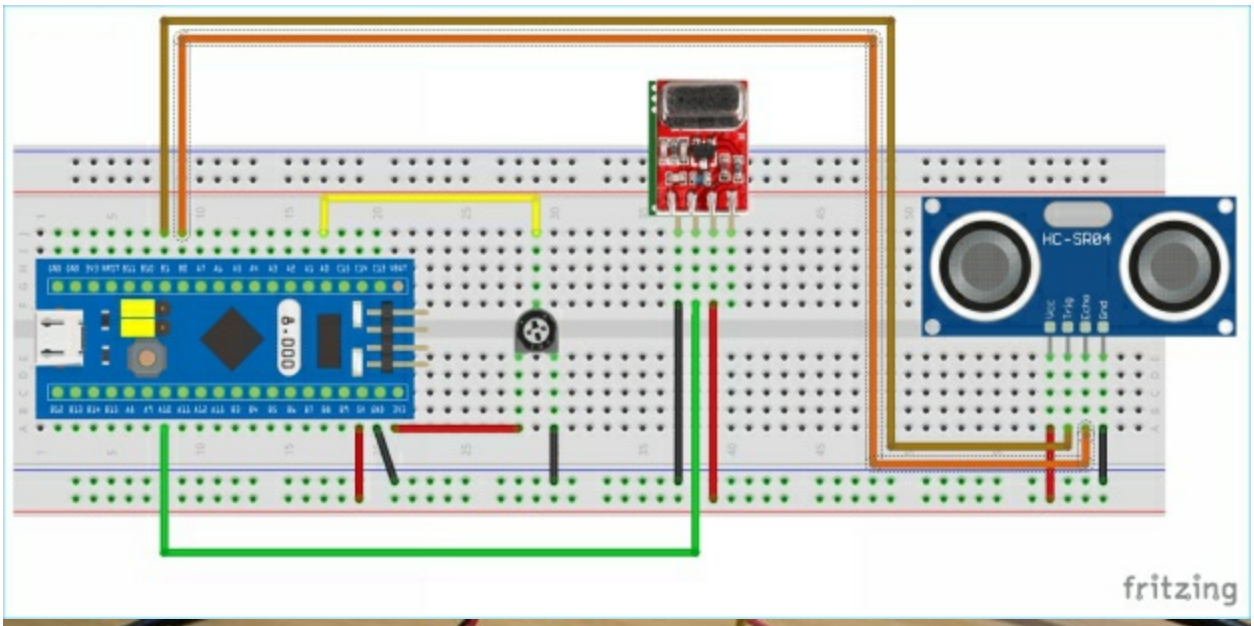
433Mhz RF Receiver	**USE**
ANT	For connecting Antenna

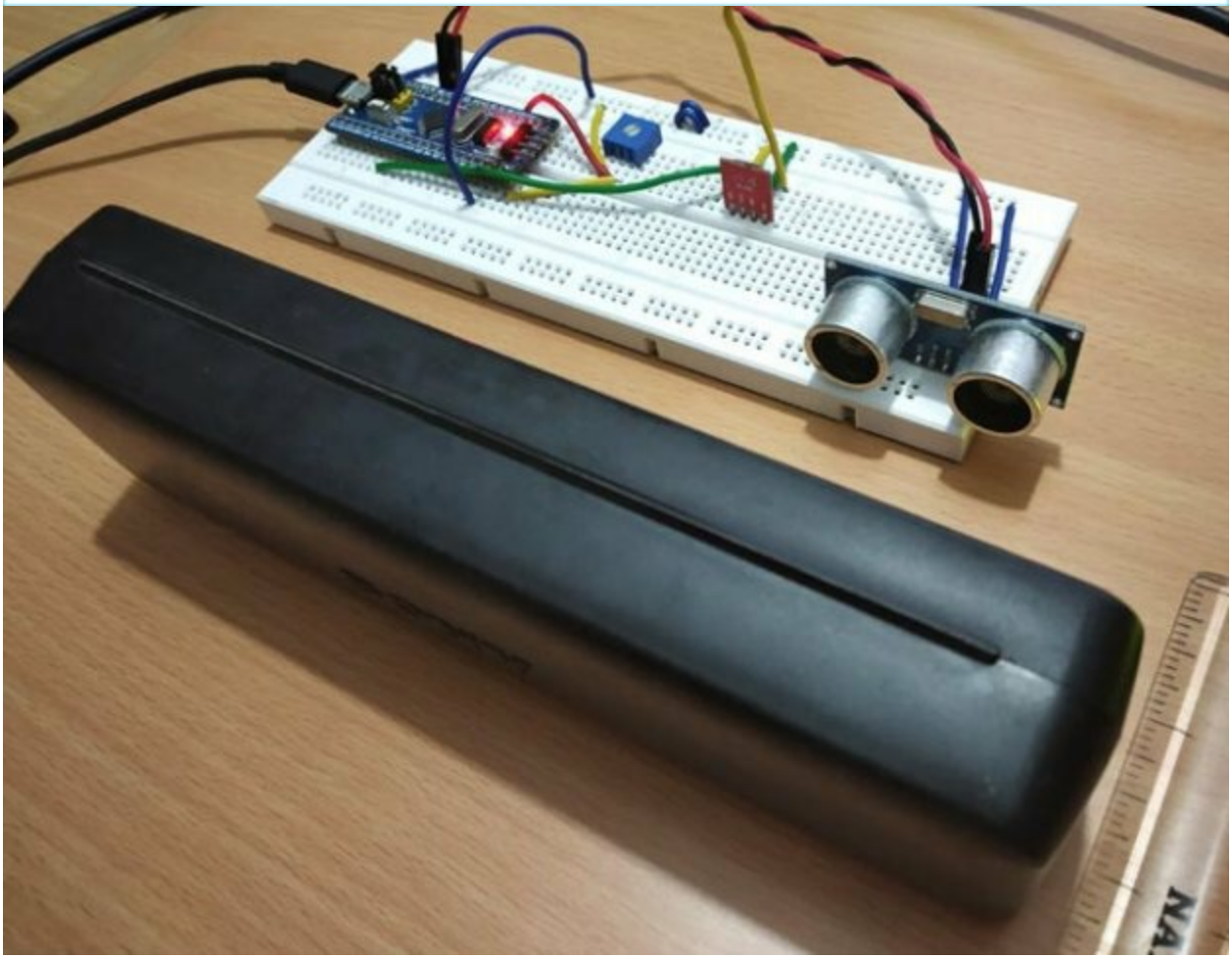
GND	GND
VDD	3.3 to 5V
DATA	Data to be received from Transmitter
CE/DO	It is also a Data pin

433 MHz Module Specifications:

- Recipient Operating Voltage: 3V to 5V
- Transmitter Operating Voltage: 3V to 5V
- Working recurrence: 433 MHz
- Transmission Distance: 3 meters (without recieving wire) to 100 meters (most extreme)
- Balancing Technique: ASK (Amplitude move keying)
- Information Transmission speed: 10Kbps

Circuit Diagram of RF Transmitter with STM32F103C8





Circuit Connections between RF Transmitter and STM32F103C8:

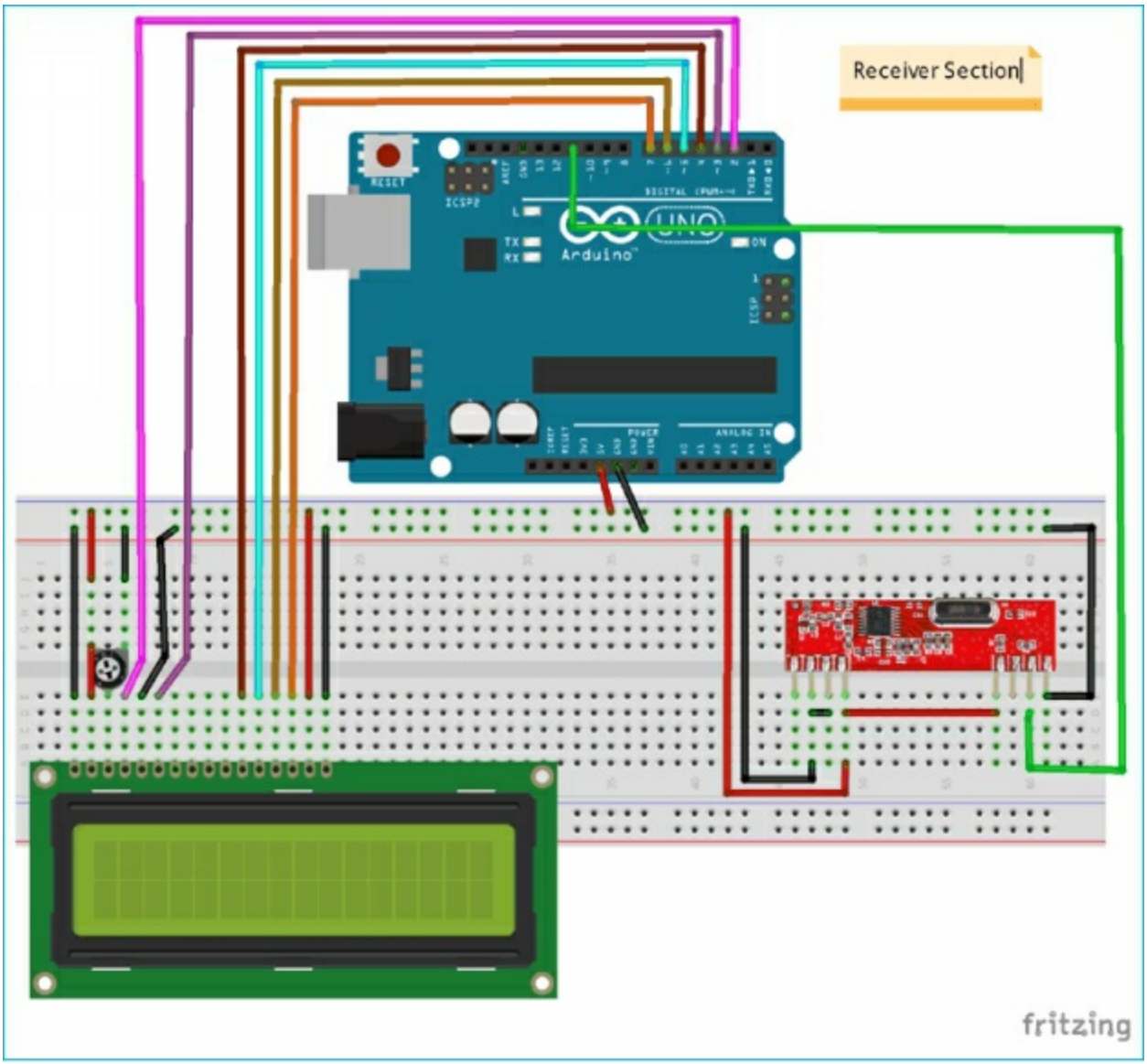
STM32F103C8	RF Transmitter
5V	VDD
GND	GND
PA10	DATA
NC	ANT

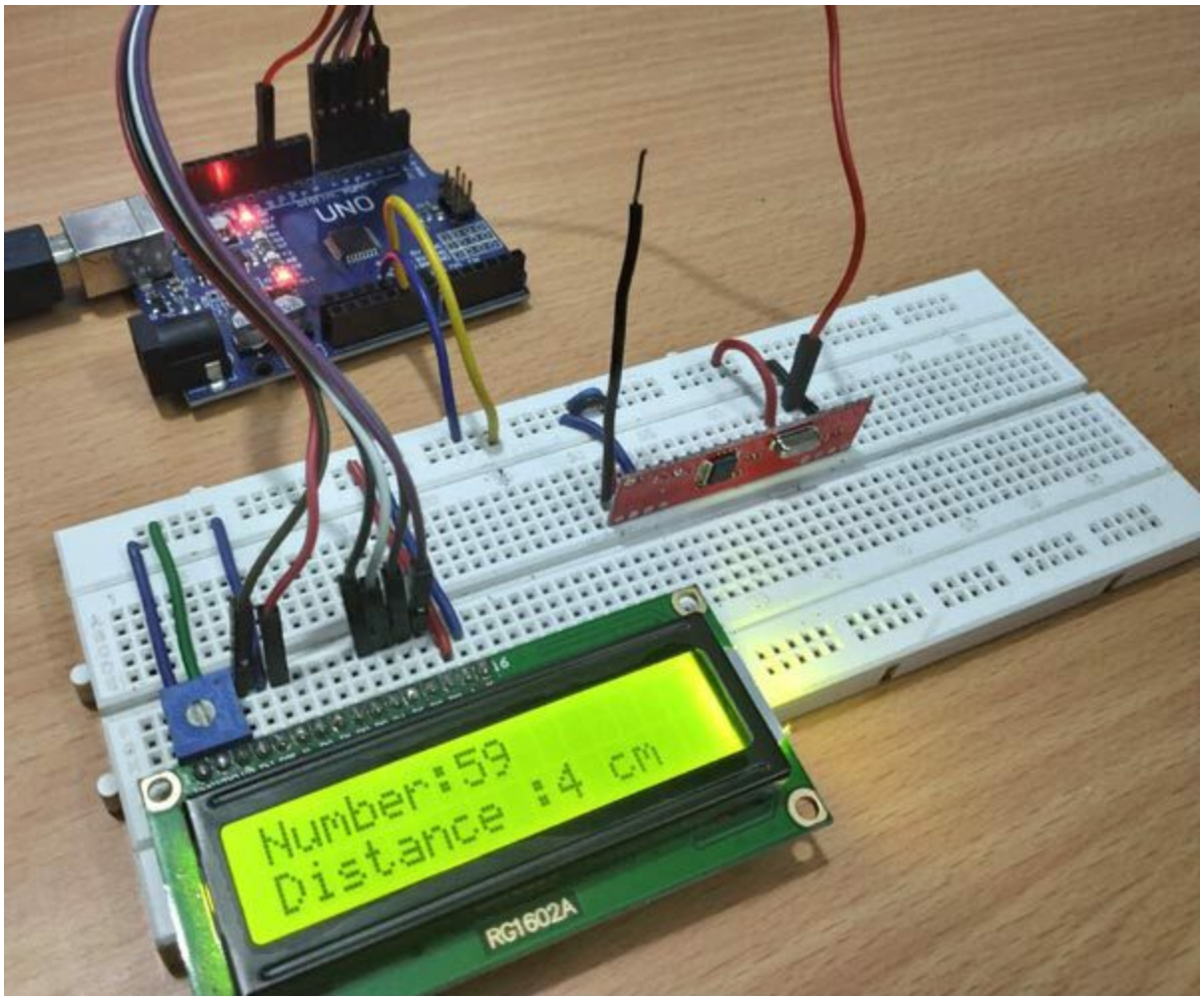
Circuit Connections between Ultrasonic Sensor and STM32F103C8:

STM32F103C8	Ultrasonic Sensor (HC-SR04)
5V	VCC
PB1	Trig
PB0	Echo
GND	GND

A 10k potentiometer is associated with the STM32F103C8 to give input Analog worth (0 to 3.3V) to the Analog-to-Digital Converter pin PA0 of STM32.

Schematic Diagram of RF Receiver with Arduino Uno


Receiver Section
RESET
ICSP2
Arduino
TX
RX
ON
ICSP
POWER
ANALOG IN
fritzing



Circuit Connections among RF Receiver and Arduino UNO:

Arduino UNO	**RF Receiver**
5V	VDD
GND	GND
11	DATA
NC	ANT

Circuit Connections among 16x2 LCD and Arduino UNO:

LCD Pin Name	Arduino UNO Pin Name
Ground (Gnd)	Ground (G)
VCC	5V
VEE	Pin from Centre of Potentiometer for Contrast
Register Select (RS)	2
Read/Write (RW)	Ground (G)
Enable (EN)	3
Data Bit 4 (DB4)	4
Data Bit 5 (DB5)	5
Data Bit 6 (DB6)	6
Data Bit 7 (DB7)	7
LED Positive	5V
LED Negative	Ground (G)

The coding will be clarified in short beneath. There will be two pieces of the sketch where initial segment will be transmitter segment and another will be

beneficiary segment. To get familiar with interfacing RF module with Arduino Uno, follow the connection.

Programming STM32F103C8 for remote RF Transmission

STM32F103C8 can be customized utilizing Arduino IDE. A FTDI software engineer or ST-Link isn't expected to transfer the code to STM32F103C8. Just associate with PC across Universal Serial Bus port of STM32 along with begin programming with ARDUINO IDE. You can get the hang of Programming your STM32 in Arduino IDE by following the connection.

In the transmitter segment the separation of the article in 'cm' is estimated utilizing ultrasonic sensor and the number an incentive from (0 to 100) set utilizing potentiometer which is transmitted through RF transmitter interfaced with STM32.

First the Radiohead library is incorporated, it very well may be downloaded from here. As this library utilizes the ASK (Amplitude Shift Keying Technique) to transmit and get information. This makes the programming extremely simple. You can remember library for sketch by going into Sketch->include library->Add .zip library.

```
#include <RH_ASK.h>
```

As in this instructional exercise in the transmitter side a ultrasonic sensor is utilized to quantify the separation so the trigger and reverberation pins are characterized.

```
#define trigPin PB1

#define echoPin PB0
```

Next the item name for the RH_ASK library is set as rf_driver with the parameters, for example, speed (2000), RX pin (PA9) and TX pin (PA10).

```
RH_ASK rf_driver(2000, PA9, PA10);
```

Next the Strings variable required in this program are pronounced.

```
String transmit_number;

String transmit_distance;

String transmit;
```

Next in the void arrangement(), the article for RH_ASK rf_driver is instated.

```
rf_driver.init();
```

After that the trigger pin is set as OUTPUT pin and the PA0 (associated with potentiometer) and reverberation pin is set as INPUT pin. Sequential correspondence is start at baud pace of 9600.

```
Serial.begin(9600);

  pinMode(PA0,INPUT);

  pinMode(echoPin,INPUT);

  pinMode(trigPin,OUTPUT);
```

Next in the void circle(), frst the potentiometer esteem that is the information Analog voltage is changed over into advanced worth (ADC esteem is found). As the ADC of STM32 has 12-piece goals. Thus, the advanced worth shifts from (0 to 4096) which is mapped into (0 to 100).

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

```
int pwmvalue = map(analoginput,0,4095,0,100);
```

Next the separation is estimated utilizing ultrasonic sensor by setting the trigger high and low with a postponement of 2 microseconds.

```
digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
```

The reverberation pin detects the reflected wave back, that is the time term that activated wave is reflected back is utilized in ascertaining the division of item utilizing the recipe. Learn all the more how ultrasonic sensor ascertains separation, by following the connection.

```
long duration = pulseIn(echoPin, HIGH);
 float distance= duration*0.034/2;
```

Presently both the information number and separation estimated is changed over into string information and put away in individual string factors.

```
transmit_number= String(pwmvalue);
  transmit_distance = String(distance);
```

Both the string is included as one line and put away in string called transmit and comma "," is utilized to isolate two strings.

```
transmit = transmit_pwm + "," + transmit_distance;
```

The transmit string is changed over into character exhibit.

```
const char *msg = transmit.c_str();
```

The information is transmitted and hold up till it is sent.

```
rf_driver.send((uint8_t *)msg, strlen(msg));

rf_driver.waitPacketSent();
```

The string information sent is likewise shown in the Serial Monitor.

```
Serial.println(msg);
```

Programming Arduino UNO as RF Receiver

Arduino UNO is customized utilizing the Arduino IDE. In the collector area the information that is transmitted from the transmitter segment and got by the RF recipient module and the string information got is part into particular information (separation and number) and showed in the 16x2 LCD show.

How about we view the collector coding to sum things up:

Like in the transmitter segment first the RadiohHead library is incorporated. As this library utilizes the (Amplitude Shift Keying Technique) to transmit and get information. This makes the programming exceptionally simple.

```
#include <RH_ASK.h>
```

As LCD show is utilized here so the liquidcrystal library is likewise included.

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

What's more, the 16x2 LCD show pins associated with Arduino UNO are indicated and proclaimed utilizing lcd as item.

```
LiquidCrystal lcd(2,3,4,5,6,7);
```

Next the String information factors to store string information are proclaimed.

```
String str_receive;

String str_number;

String str_distance;
```

The article for the Radiohead library is pronounced.

```
  RH_ASK rf;
```

Presently in the void arrangement(), The LCD show is set in 16x2 mode and an invite message is shown and cleared.

```
 lcd.begin(16,2);

   lcd.print("Hello_world");

   lcd.setCursor(0,1);

   lcd.print("RF with STM32");

   delay(5000);
```

```
    lcd.clear();
```

From that point onward, the rf object is introduced.

```
rf.init();
```

Presently in the void circle(), the Array buf[] is announced with size as 7. As the information sent from transmitter has 7 including the ",". Thus, change this as indicated by the information that will be transmitted.

```
uint8_t buf[7];

    uint8_t buflen = sizeof(buf);
```

In the event that the string is accessible at the rf recipient module the if work checks the size and it executes. The rf.recv() is utilized to get information.

```
if (rf.recv(buf, &buflen))
```

The buf has the gotten string so then got string is put away in a str_receive string variable.

```
str_receive = String((char*)buf);
```

This for circle is utilized to part the got string into two on the off chance that it recognizes the ',' in the centre of two strings.

```
  for (int i = 0; i < str_receive.length(); i++)

      {
```

```
if (str_receive.substring(i, i+1) == ",")

{

str_number = str_receive.substring(0, i);

str_distance = str_receive.substring(i+1);

break;

}
```

Two roast clusters for two qualities are pronounced and the String that is part into two is put away in regarded exhibit by changing over string into character exhibit.

```
char numberstring[4];

   char distancestring[3];

   str_distance.toCharArray(distancestring,3);

   str_number.toCharArray(numberstring,3);
```

After that convert the character exhibit into whole number utilizing atoi()

```
int distance = atoi(distancestring);

   int number = atoi(numberstring);
```

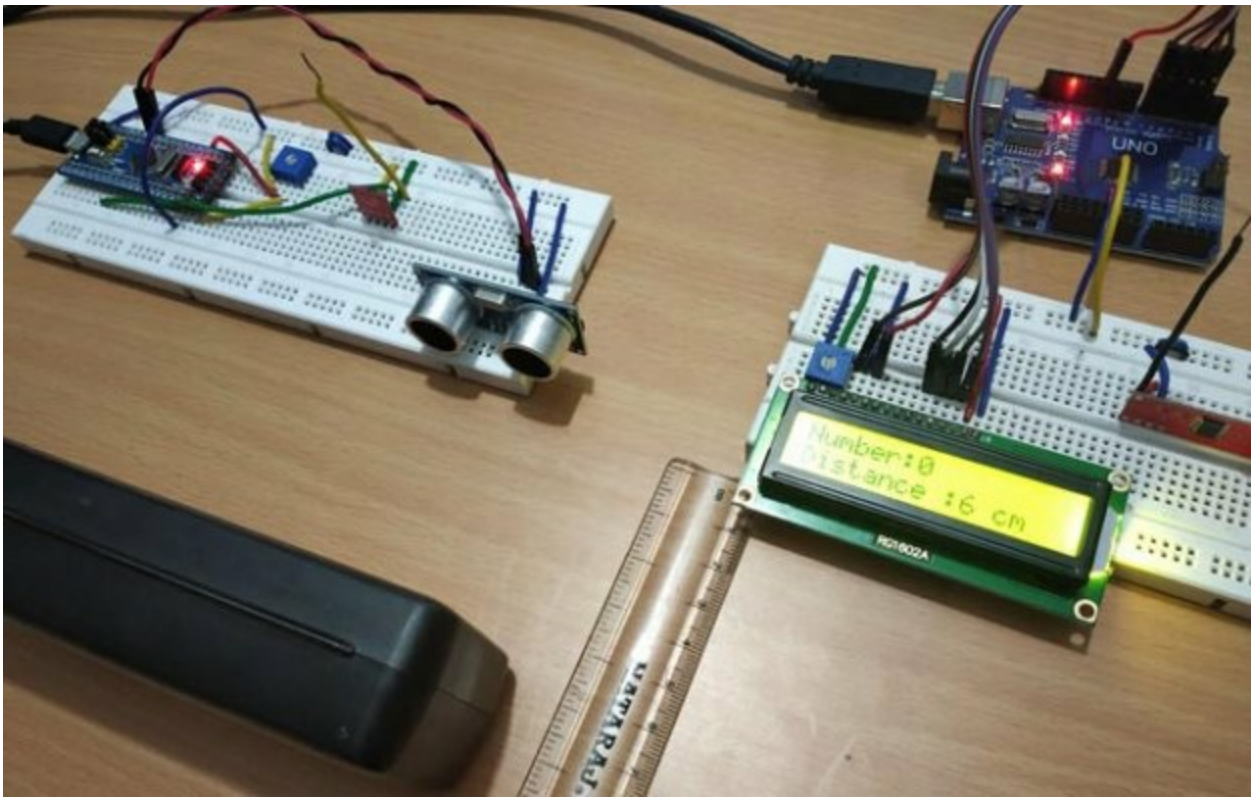
Subsequent to changing over into whole number qualities the qualities separation and number is shown in 16x2 LCD show

```
lcd.setCursor(0,0);
        lcd.print("Number:");
        lcd.print(number);
        lcd.setCursor(0,1);
        lcd.print("Distance :");
        lcd.print(distance);
        lcd.print(" cm");
```

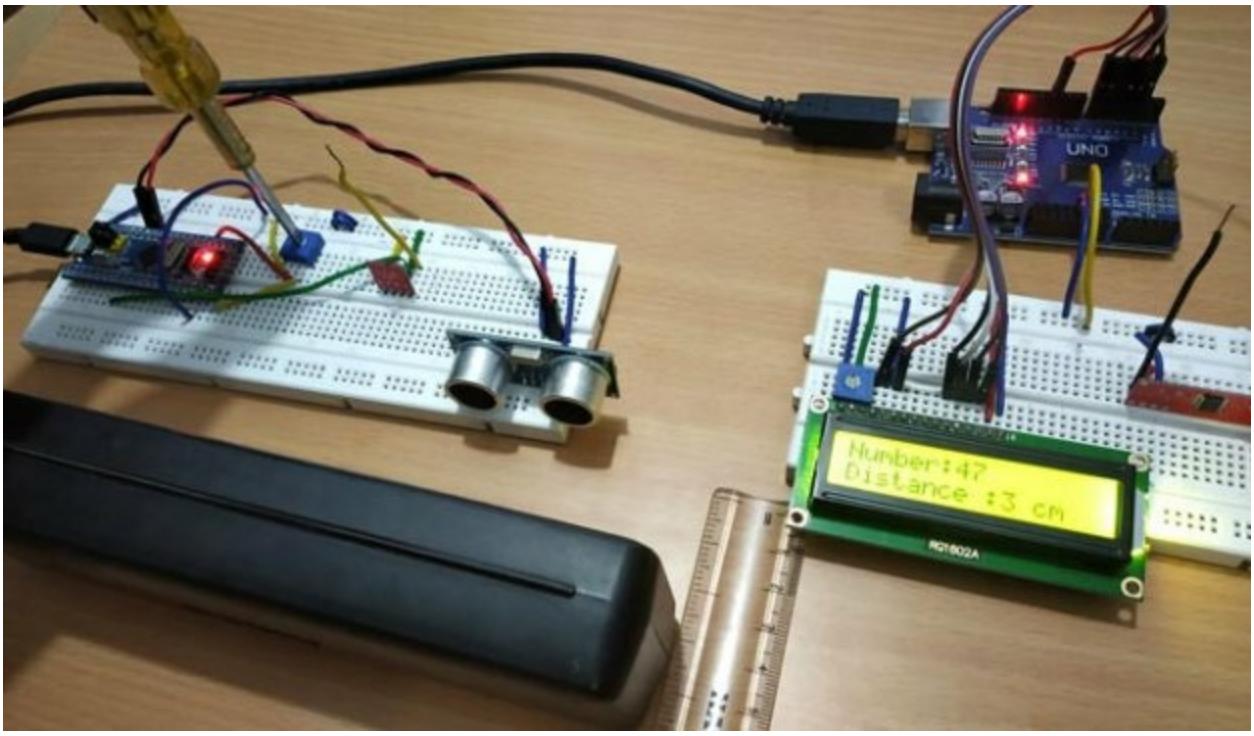
Subsequent to transferring both the codes for example transmitter along with recipient in the STM32 and Arduino UNO individually, the information, for example, number and article separation estimated utilizing the STM32 is transmitted to the RF beneficiary by means of RF Transmitter and the qualities got are shown in the LCD show remotely.

Testing STM 32 based RF Transmitter along with Receiver

1. At the point when number at 0 and the separation of item is at 6cm.



2. In this point when number 47 and separation of item is at 3cm.



Code

Transmitter code (STM32F103C8):

```
//433MHZ RF Trasmitter with STM32F103C8
//Transmitter Code
#include <RH_ASK.h>                    //RadioHead library
#define trigPin PB1                    //Sets the Trigpin of Ultrasonic sensor as PB1
#define echoPin PB0                    //Sets the echoPin of Ultrasonic sensor as PB0
RH_ASK rf_driver(2000, PA9, PA10);     //Sets Pin PA9 as receiver and PA10 as transmitterand 2000 as Speed
String transmit_pwm;                   //Strings to store string value
String transmit_distance;
String transmit;
```

```
void setup()
{
  // Initialize ASK Object
  rf_driver.init();
  Serial.begin(9600);
  pinMode(PA0,INPUT);
  pinMode(echoPin,INPUT);
  pinMode(trigPin,OUTPUT);
}
void loop()
{
  int analoginput = analogRead(PA0);                // ADC value from pin PA0 connected to Potentiometer
  int pwmvalue = map(analoginput,0,4096,0,100);     // Converts 0 to 4096 into 0 to 100
  digitalWrite(trigPin, LOW);                       //Makes TrigPin of Ultrasonic LOW
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);                      //Makes TrigPin of Ultrasonic HIGH
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);                       //Makes TrigPin of Ultrasonic LOW
  long duration = pulseIn(echoPin, HIGH);           //Receives the Echo signal reflected
  float distance= duration*0.034/2;                 //Calculates distance in CM of object
```

```
   transmit_pwm = String(pwmvalue);                    //Convert value into string
   transmit_distance = String(distance);               //Convert value into string
   transmit = transmit_pwm + "," + transmit_distance;  //Adds two String in one line
   const char *msg = transmit.c_str();                 //
   rf_driver.send((uint8_t *)msg, strlen(msg));        //Sends the String
   rf_driver.waitPacketSent();
   Serial.println(msg);                                //Serial Print value for debug
   delay(1000);
}
```

Receiver Code (Arduino UNO):

```
//Receiver Arduino Code
//433MHZ RF with STM32F103C8 as Transmitter
#include <RH_ASK.h>                 //Includes RadioHead Library
#include <LiquidCrystal.h>          //Includes the LCD display Library
LiquidCrystal lcd(2,3,4,5,6,7);     //Initialize lcd with Pins connected to Arduino
String str_receive;                 //Strings to Store Value
String str_number;
String str_distance;
RH_ASK rf;                          //rf as object for RG_ASK
void setup()
{
   lcd.begin(16,2);                 //Lcd set as 16x2 Mode
   lcd.print("Hello_world");        //Display Welcome message
```

```
  lcd.setCursor(0,1);
  lcd.print("RF with STM32");
  delay(5000);
  lcd.clear();
  rf.init();                    //Initialize rf Object
}
void loop()
{
  uint8_t buf[7];
  uint8_t buflen = sizeof(buf);
  if (rf.recv(buf, &buflen))
  {
    str_receive = String((char*)buf);                    // Receive String from the Transmitter
    for (int i = 0; i < str_receive.length(); i++)          // Split string into two string
    {
    if (str_receive.substring(i, i+1) == ",")
    {
    str_number = str_receive.substring(0, i);
    str_distance = str_receive.substring(i+1);
    break;
    }
  }
    char numberstring[4];
    char distancestring[3];
```

```
    str_distance.toCharArray(distancestring,3);            //Convert String into Char Array
    str_number.toCharArray(numberstring,3);
    int distance = atoi(distancestring);                   //Convery Array into integer value
    int number = atoi(numberstring);
    lcd.setCursor(0,0);
    lcd.print("Number:");
    lcd.print(number);                                     //Display number value at LCD display
    lcd.setCursor(0,1);
    lcd.print("Distance :");
    lcd.print(distance);                                   //Display distance value at LCD display
    lcd.print(" cm");
    delay(1500);
    lcd.clear();
}
}
```

◆ ◆ ◆

7. SEND/RECEIVE SMS WITH STM32F103C8 AND SIM800C GSM MODULE

GSM Modules are much of the time utilized in IoT ventures since it can send and get information remotely. The Global System for Mobile Modules doesn't have a lot of conditions like the Wi-Fi modules. While different remote modules such Wi-Fi otherwise Zigbee have cons, for example, short range along with cost, the Global System for Mobile Module have the unwavering quality, long range Since only one SIM card is required with legitimate arrangement. In case the GSM modules can't send an immense measure of information created by sensors, it is perfect for applications where little sensible measure of information to be sent.

In this instructional exercise, such GSM module will be interfaced with STM32F103C8 ARM microcontroller to send and get Text Messages (SMS) from the cell portable number arranged in the program. To show the messages got and sent, one 16x2 LCD will be utilized with two Push Buttons which will start sending and getting of instant messages in the wake of squeezing the relative catch.

There are as of now a few GSM ventures are accessible interfaced with various sort of microcontrollers. You can proceed to check all the GSM

Projects here, and attempt DIY by referencing past instructional exercises on various GSM modules, for example, SIM900, SIM900A, SIM800, and so forth. A portion of the undertakings with GSM modules are given beneath:

- Call and Message utilizing Arduino and GSM Module
- Remote Notice Board utilizing GSM and Arduino
- Interfacing Global System for Mobile Module with AVR Microcontroller: Send and Receive Messages
- Global System for Mobile Module Interfacing with 8051 Microcontroller
- Arduino Based Vehicle Accident Alert System utilizing GPS, GSM and Accelerometer

Segments Required

- STM32F103C8 Cortex-M3 Microcontroller
- GSM Module (SIM800C is utilized in the instructional exercise)
- 16x2 LCD Display
- Press Buttons (2)
- 10k Potentiometer
- Breadboard
- Associating Wires

What is SIM800C GSM Module?

SIM800C is a generally utilized GSM Module with a sequential interface modem which runs in the centre of 3.4V-4.4V Voltage level. SIM800C is a Quad-band GSM/GPRS Module which is utilized in inserted applications where the remote information move is required. SIM800C takes a shot at 850/900/1800/1900MHz. It can likewise get and transmit Voice Call, SMS with low force utilization. The module is constrained by utilizing AT orders. It underpins one SIM card interface and has UART (TX and RX) sticks alongside one RS232 Serial Protocol that can be used to interface with various microcontrollers in implanted applications.

Driving SIM800C GSM Module

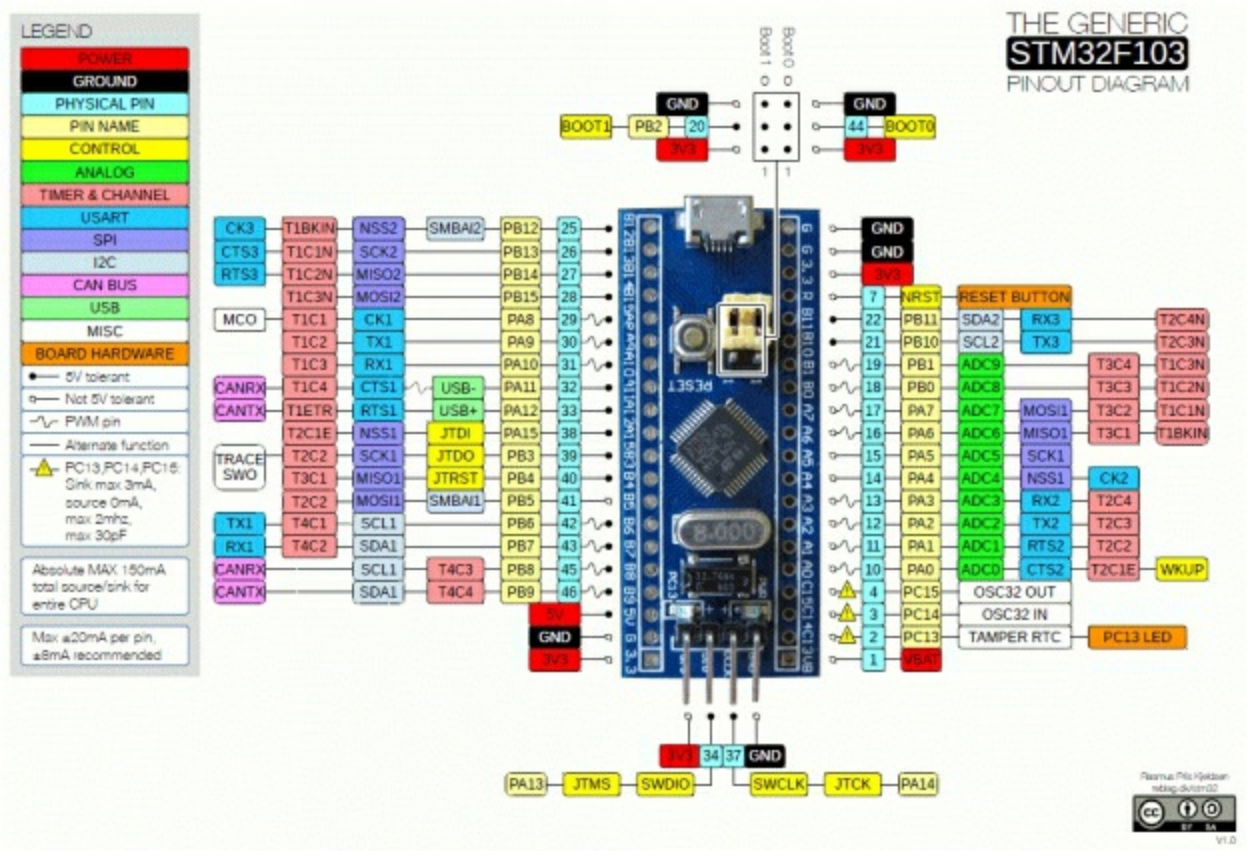
A DC Power connector of 12V is utilized to control the SIM800C GSM module.

Embeddings SIM in SIM800C GSM Module

A SIM card is embedded at the rear of the SIM800C GSM module. Note that the SIM800C doesn't bolster 4G, so don't embed a 4G SIM Card.

Interfacing GSM SIM800C with STM32F103C8

So as to interface SIM800C with STM32F103C8 microcontroller, the UART Serial port will be utilized which is an equipment sequential interface of the STM32F103C8. The underneath picture demonstrates the UART sticks in STM32F103C8 which are A9 and A10.



The STM32F103C8 has three UART interface for associating three outer sequential peripherals.

AT Commands

The AT-orders will be utilized to get to the elements of GSM Module, for example, sending and getting Voice Calls, Text Messages. A portion of the AT orders are given beneath which will be significant in this instructional exercise and will be utilized regularly.

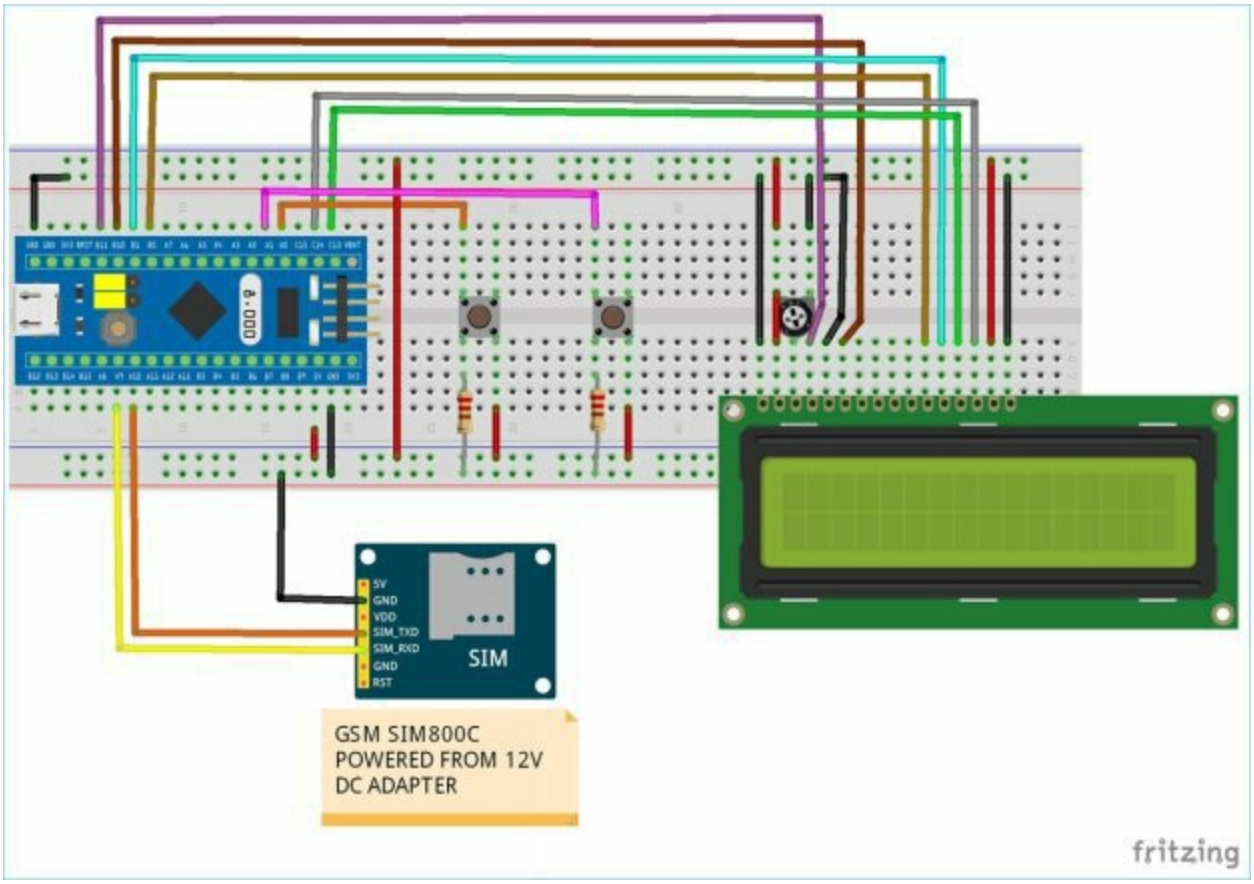
AT	Replies with OK for Acknowledgement
AT+CPIN?	Check signal Quality

AT+COPS?	Find service provider name
ATDXXXXXXXXXX;	Call to the specific number, ends with semi-colon,replace X with mobile number
AT+CNUM	Find the number of SIM card (might not work for some SIM)
ATA	Answer the Incoming Call
ATH	Hang off the current Incoming call
AT+COLP	Show incoming call number
AT+VTS=(number)	Send DTMF number. You can use any number on your mobile keypad for (number)
AT+CMGR	AT+CMGR=1 reads message at first position
AT+CMGD=1	Delete message at first position
AT+CMGDA=”DEL ALL”	Delete All messages from SIM
AT+CMGL=”ALL”	Read all messaged from SIM
AT+CMGF=1	Set SMS configuration. “1”

	is for text only mode
AT+CMGS = “+91 XXXXXXXXXX” >HelloworldText<Ctrl+z>	Sends SMS to a particular number here XXXXXXXXXX. When you see “>” start entering the text. Press Ctrl+Z to send the text.
AT+CNMI=2,2,0,0,0	To receive Live SMS
AT+CGATT?	To check for internet connection on SIM card
AT+CIPSHUT	To close TCP connection, meaning to disconnect form internet
AT+CSTT = “APN”,”username”,”Pass”	Connect to GPRS with your APN and Pass key. Can be obtained from Network Provider.
AT+CIICR	Check if SIM card has data pack
AT+CIFSR	Get IP of the SIM network
AT+CIPSTART = “TCP”,”SERVER IP”,”PORT”	Used to set a TCP IP connection
AT+CIPSEND	This command is used to send data to server

Circuit Diagram

Associations for interfacing GSM with STM32 is appeared in the circuit outline beneath.



Circuit associations between STM32F103C8 and GSM SIM800C

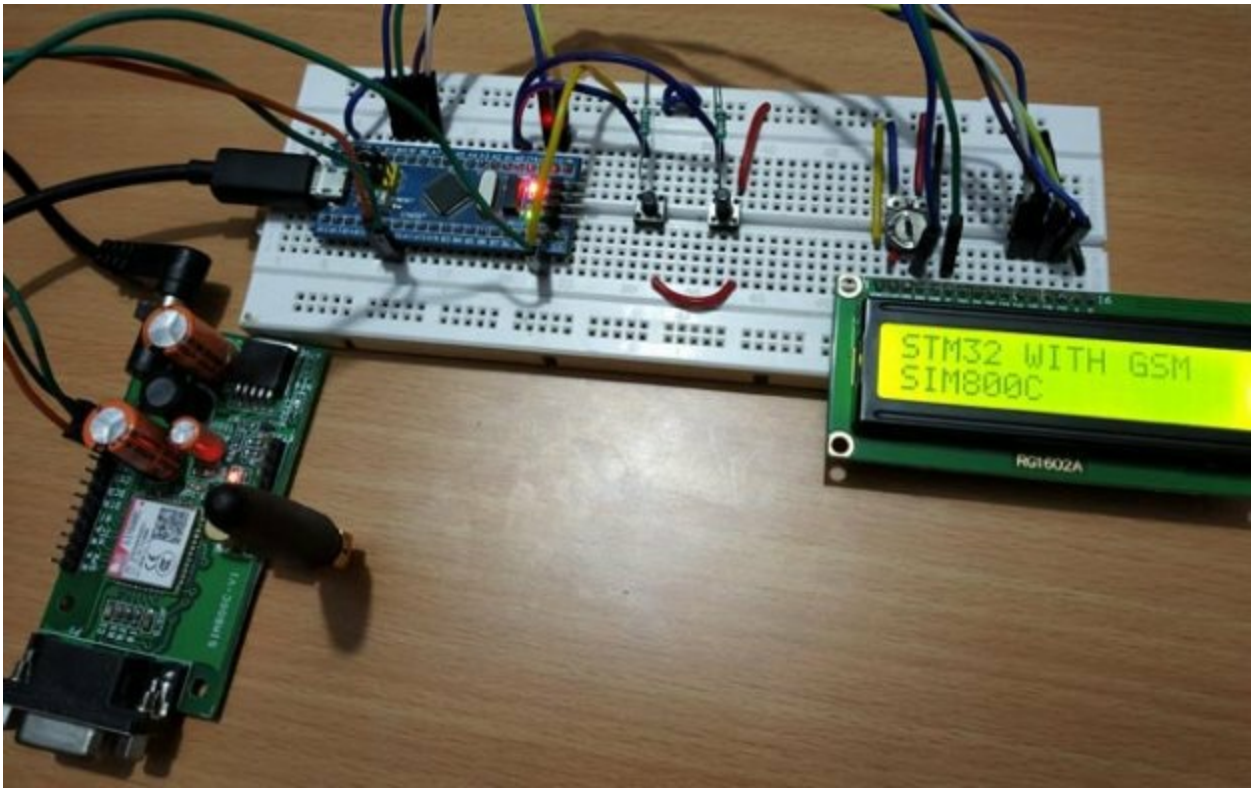
STM32F103C8	**GSM SIM800C**
PA9 (TX)	RX
PA10 (RX)	TX
GND	GND

Circuit associations between STM32F103C8 and 16x2 LCD

LCD Pin No	LCD Pin Name	STM32 Pin Name
1	Ground (Gnd)	Ground (G)
2	VCC	5V
3	VEE	Pin from Centre of Potentiometer for contrast
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)

Two Push catches with Pull down resistor of 10k is associated with the pins PA0 and PA1 of STM32 microcontroller. The total arrangement will look lik underneath:



Programming STM32F103C8 Microcontroller for GSM interfacing

STM32F103C8 microcontroller can be modified utilizing ARDUINO IDE. In this instructional exercise, FTDI or ST-LINK developer isn't expected to program the STM32. For transferring code to STM32F103C8, just attachment microUSB port to STM32, and USB port to PC by utilizing USB link and begin composing code in ARDUINO IDE. If there should be an occurrence of any uncertainty follow our past instructional exercises on the most proficient method to program STM32 without utilizing any outer FTDI or ST-LINK Programmer.

The total code can be found toward the finish of this instructional exercise.

Start with including important libraries for peripherals utilized in this instructional exercise. Likewise characterize the pin setup of LCD.

```
#include <LiquidCrystal.h>

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

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

Next, initialise Liquid Crystal Display choosing the capacity lcd.begin(16,2); where (16,2) tells the LCD type, for example, 16x2 square LCD show. Simply print a message to troubleshoot that if LCD is giving Output and interfaced appropriately.

```
lcd.begin(16,2);

lcd.print("STM32 WITH GSM");

lcd.setCursor(0,1);
```

```
lcd.print("SIM800C");
```

Characterize the Push button information course as Input Mode and characterize the pin number utilized.

```
pinMode(PA0,INPUT);

pinMode(PA1,INPUT);
```

Select the baud rate utilized for sequential correspondence.

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

There are 2 capacities in the code one to send message and other one to get message. Definite clarification is given underneath.

SendMessage

This capacity is used to send Short Message Service to a number. AT order is sent to GSM module showing the message content mode and on which number the message should be sent.

The beneath explanation sets the Global System for Mobile module in the Text Mode by sending AT order (AT+CMGF=1).

```
Serial1.println("AT+CMGF=1");  //Sets the GSM Module in Text Mode
```

After that AT order (AT+CMGS) showing send a SMS with the versatile number is sent to GSM module through serial1 port.

```
Serial1.println("AT+CMGS=\"+91XXXXXXXXXX\"\r"); // Replace x with mobile number
```

The messages are sent utilizing AT order after each one second and afterward CLRL+Z should be sent so an ASCII code of CTRL+Z is sent through Serial1 port to GSM module.

```
delay(1000);

Serial1.println("Hi Friend from GSM Module"); // The SMS text you want to send

Serial1.println((char)26);// ASCII code of CTRL+Z
```

At that point the "SMS sent" is shown in the 16X2 LCD showed.

```
lcd.print("SMS SENT");
```

ReceiveMessage

In this capacity, the messages are gotten and imprinted in the LCD show screen.

```
Serial1.println("AT+CNMI=2,2,0,0,0");
```

In this way, to get live messages the above AT order is utilized.

In the wake of getting the SMS, it contains a string accessible at the serial1 port which additionally have different information in it, for example, the time, date along with etc. In this way, after event of the 6th twofold statement (") the rest everything is the SMS gotten. Subsequently the other data is precluded and the rest of the data that is the gotten message is shown in the 16x2 LCD show.

```
while(1)
```

```
{
        if(Serial1.available())
         {
            do
              {
                 while ( !Serial1.available() );
              } while ( '"' != Serial1.read() );
            do
              {
                 while ( !Serial1.available() );
              } while ( '"' != Serial1.read() );
            do
              {
                 while ( !Serial1.available() );
              } while ( '"' != Serial1.read() );
            do
              {
                while ( !Serial1.available() );
```

```
                } while ( '"' != Serial1.read() );
               do
                 {
                    while ( !Serial1.available() );
                 } while ( '"' != Serial1.read() );
               do
                {
                   while ( !Serial1.available() );
                 } while ( '"' != Serial1.read() );
                 while ( !Serial1.available() );
                 receive = Serial1.read();
                 while ( !Serial1.available() );
                 receive = Serial1.read();
lcd.clear();
while(1)
{
    while ( !Serial1.available() );
    receive = Serial1.read();
```

```
        if ( receive == '\r' )
                break;
    else
            lcd.write(receive);
    }
}
}
```

Demo to Send along with Receive Text Message utilizing STM32

1. To send Text Message, just press the Left Push button. The SMS will be sent to the versatile number entered in the code.

2. To get Text Message, just press the Right Push button and the SMS will be gotten and will be shown in the LCD show screen.

So's everything about interfacing GSM module with STM32F103C8 ARM microcontroller.

Code

```
#include <LiquidCrystal.h>                    //Library for LCD display

const int rs = PB11, en = PB10, d4 = PB0, d5 = PB1, d6 = PC13, d7 = PC14;      //Pins that are connected between LCD and STM32

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

int receive = 0;

void setup()

{


  lcd.begin(16,2);          //LCD set at 16x2 mode
```

```
 pinMode(PA0,INPUT);       //Push buttons as INPUT pins
pinMode(PA1,INPUT);


 lcd.print("STM32 WITH GSM");   //Display Welcome message
lcd.setCursor(0,1);
lcd.print("SIM800C");


 Serial1.begin(9600);   // Setting the baud rate of GSM Module
delay(1000);
lcd.clear();
}
void loop()
{


 lcd.clear();
lcd.setCursor(0,0);
lcd.print("S to Send sms");
lcd.setCursor(0,1);
lcd.print("R to Receive sms");
delay(100);



 int a = digitalRead(PA0);          //Read status of the push buttons
int b = digitalRead(PA1);


 if (a == 1)                        // Depeding upon which push button is pressed
the respected function is called
    {
      SendMessage();
```

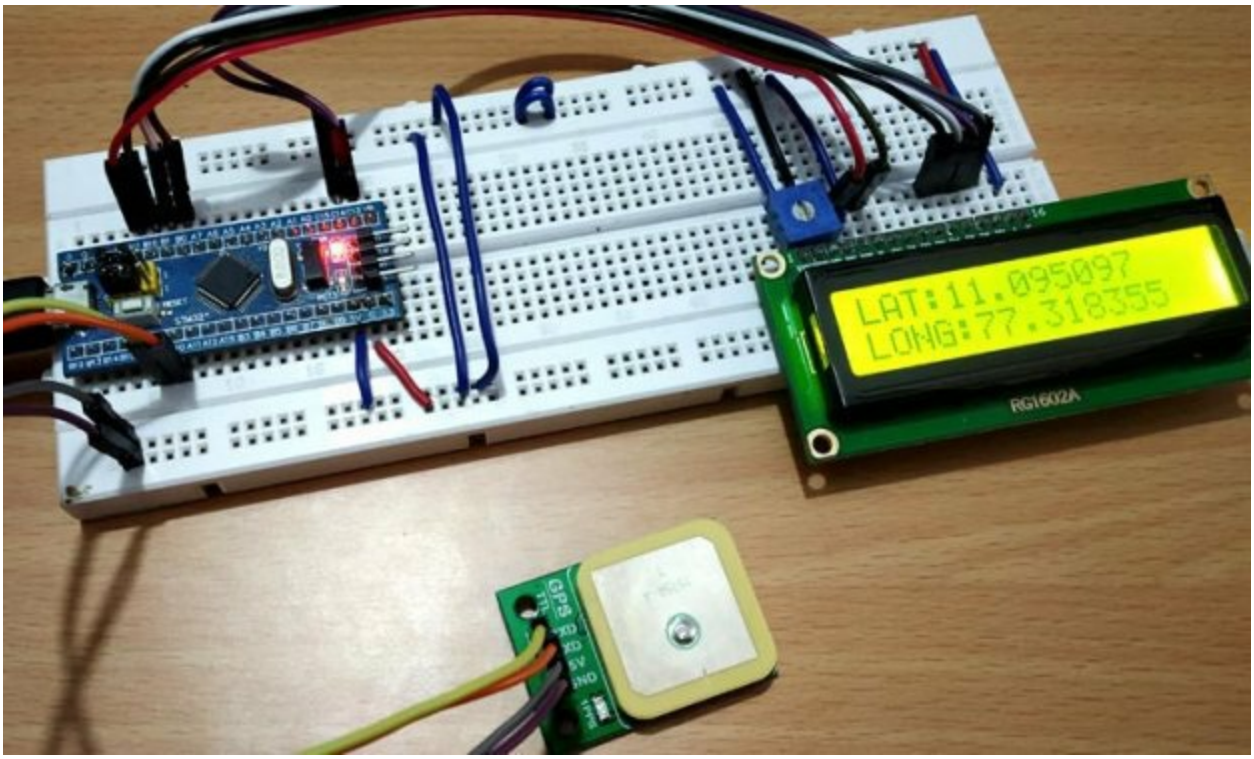
```
      }


else if( b == 1)
     {
       RecieveMessage();
     }


}


 void SendMessage()      //Function to Send Message
{
  lcd.clear();


   lcd.print("Sending sms");
  delay(1000);
  Serial1.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);
  Serial1.println("AT+CMGS=\"+91XXXXXXXXXX\"\r"); // Replace x with mobile number
  delay(1000);
  Serial1.println("Hi Friend from GSM Module"); // The SMS text you want to send
  delay(100);
  Serial1.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
  lcd.clear();
  lcd.print("SMS SENT");
  delay(1000);
}
 void RecieveMessage()         //Function to Receive Message
```

```
{


  lcd.clear();
 lcd.print("Receiving sms");
 Serial1.println("AT+CNMI=2,2,0,0,0"); // AT Command to recieve a live SMS
 delay(1000);


  while(1)
 {
        if(Serial1.available())
         {
            do
              {
                 while ( !Serial1.available() );
              } while ( "" != Serial1.read() );
            do
              {
                 while ( !Serial1.available() );
              } while ( "" != Serial1.read() );
            do
              {
                 while ( !Serial1.available() );
              } while ( "" != Serial1.read() );
            do
              {
                while ( !Serial1.available() );
              } while ( "" != Serial1.read() );
```

```
            do

                  {
                while ( !Serial1.available() );

              } while ( "" != Serial1.read() );
            do
             {

                    while ( !Serial1.available() );

              } while ( "" != Serial1.read() );
              while ( !Serial1.available() );
              receive = Serial1.read();
              while ( !Serial1.available() );
              receive = Serial1.read();

  lcd.clear();
  while(1)
  {
     while ( !Serial1.available() );
     receive = Serial1.read();
     if ( receive == '\r' )

              break;

        else
```

```
            lcd.write(receive);
    }
}
}
}
```

8. INSTRUCTIONS TO USE GLOBAL POSITIONING SYSTEM MODULE WITH STM32F103C8 TO GET LOCATION COORDINATES



GPS represents Global Positioning System along with used to recognize the Latitude along with Longitude of any area on the Earth, with careful UTC time. This gadget gets the directions from the satellite for every single second, with time along with date. GPS offers extraordinary exactness and furthermore gives other information other than position facilitates.

As a whole we understand that GPS is a helpful gadget and normally utilized in cell phones along with other compact gadgets for following the area. It has wide scope of uses in each field from calling the taxi at your home to follow the elevation of planes. Here are some valuable GPS related undertakings, we assembled already:

- Vehicle Tracking System
- GPS Clock
- Mishap Detection Alert System
- Raspberry Pi GPS Module Interfacing Tutorial

- Interfacing Global Positioning System Module with PIC Microcontroller

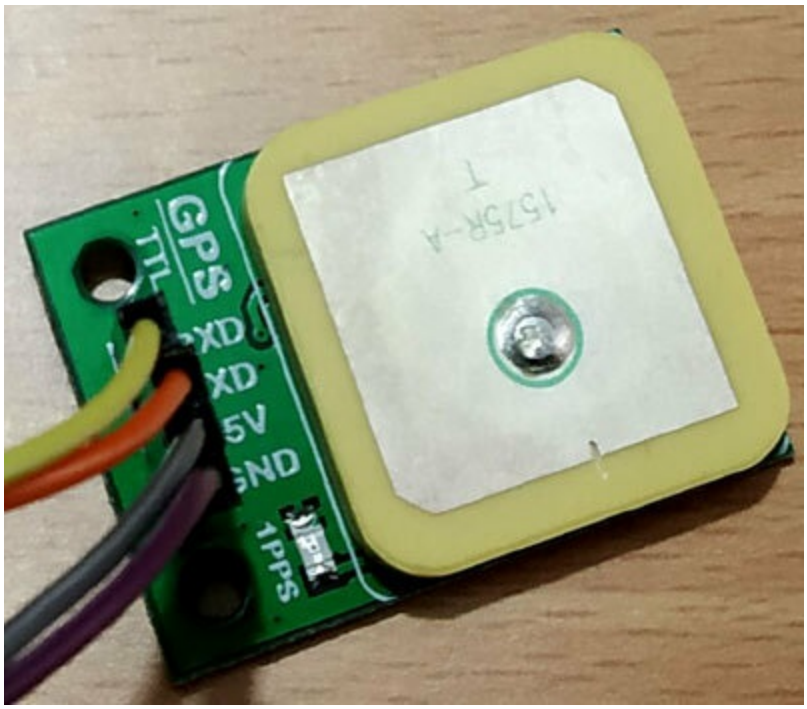
Here in this instructional exercise, we will Interface a Global Positioning System module with STM32F103C8 microcontroller to discover the area organizes along with demonstrate on 16x2 LCD show.

Segments Required

- STM32F103C8 Microcontroller
- GPS Module
- 16x2 LCD show
- Breadboard
- Associating Wires

GPS Module

It's a GY-NEO6MV2 XM37-1612 GPS Module. This GPS module has four pin +5V, GND, TXD and RXD. It imparts utilizing the Serial pins and can be effectively interfaced with the Serial port of the STM32F103C8.



GPS module sends the information in NMEA position (see the screen capture beneath). NMEA group comprise a few sentences, wherein we just need one sentence. This sentence begins from $GPGGA and contains the directions, time and other helpful data. This GPGGA is alluded to Global Positioning System Fix Data. Find out about Reading GPS information and its strings here.

The following is one example $GPGGA String, alongside its portrayal:

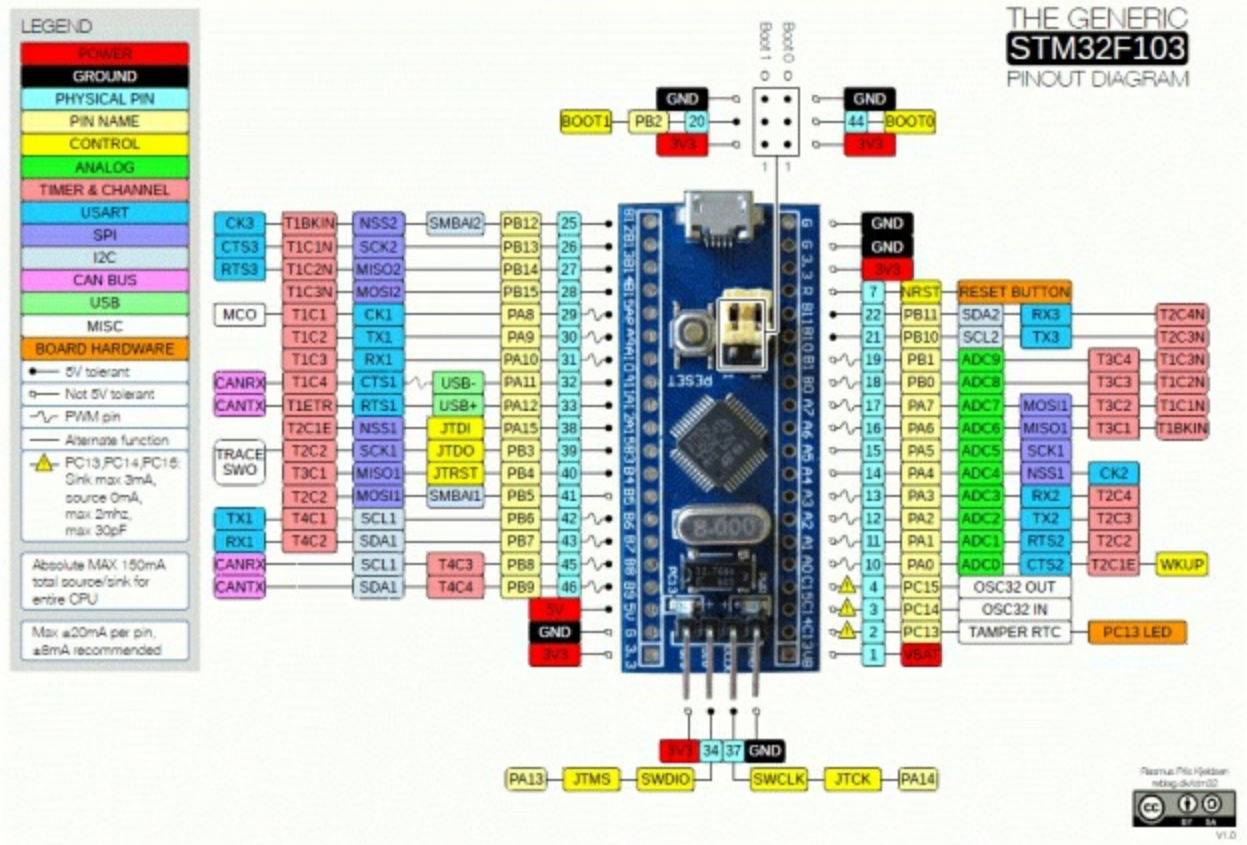
$GPGGA,104534.000,7791.0381,N,06727.4434,E,1,08,0.9,510.4,M,43.9,M,,

$GPGGA,HHMMSS.SSS,latitude,N,longitude,E,FQ,NOS,HDP,altitude,M,he information

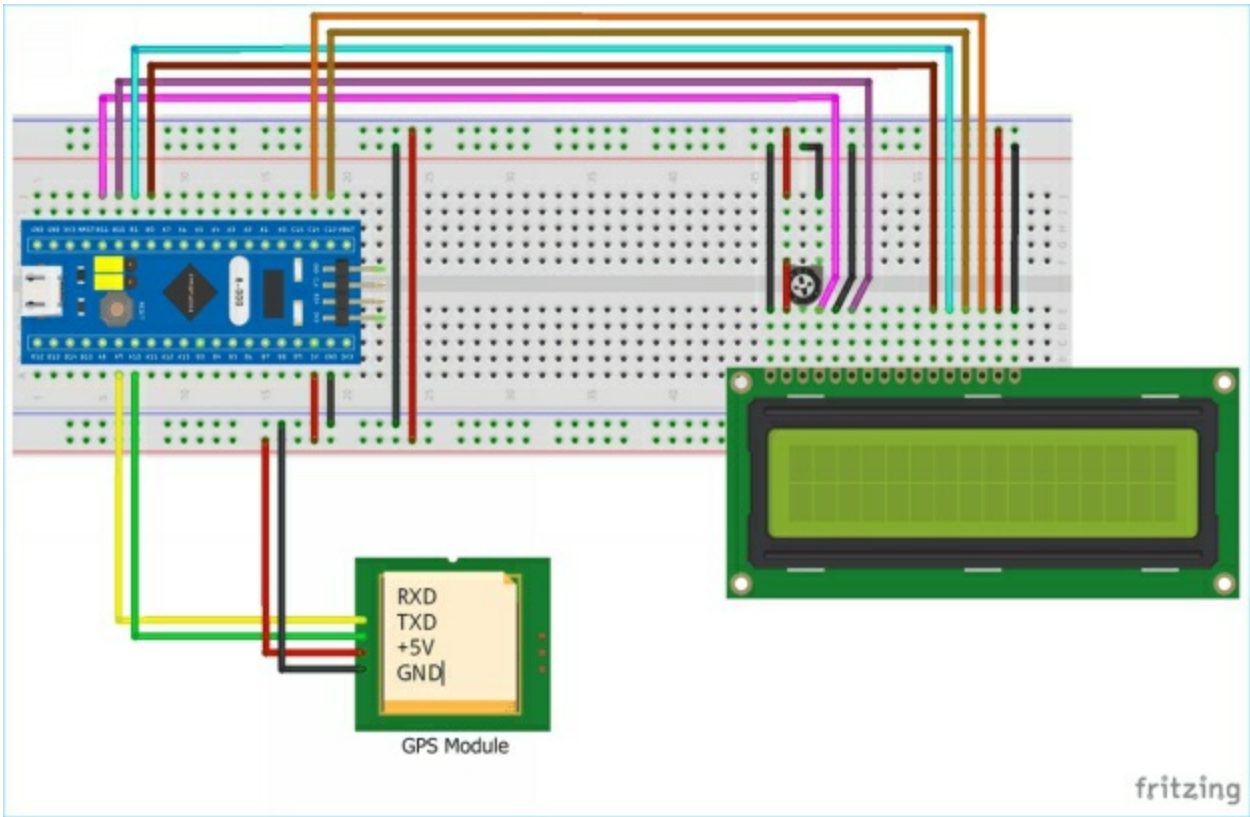
Be that as it may, here in this instructional exercise, we are utilizing a TinyGPSPlus GPS library which separates all the necessary data from the NMEA sentence, and we simply need to compose a straightforward line of code to get the scope and longitude, which we will see later in the instructional exercise.

Pin out of STM32F103C8

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



Circuit Joints among GPS module and STM32F103C8

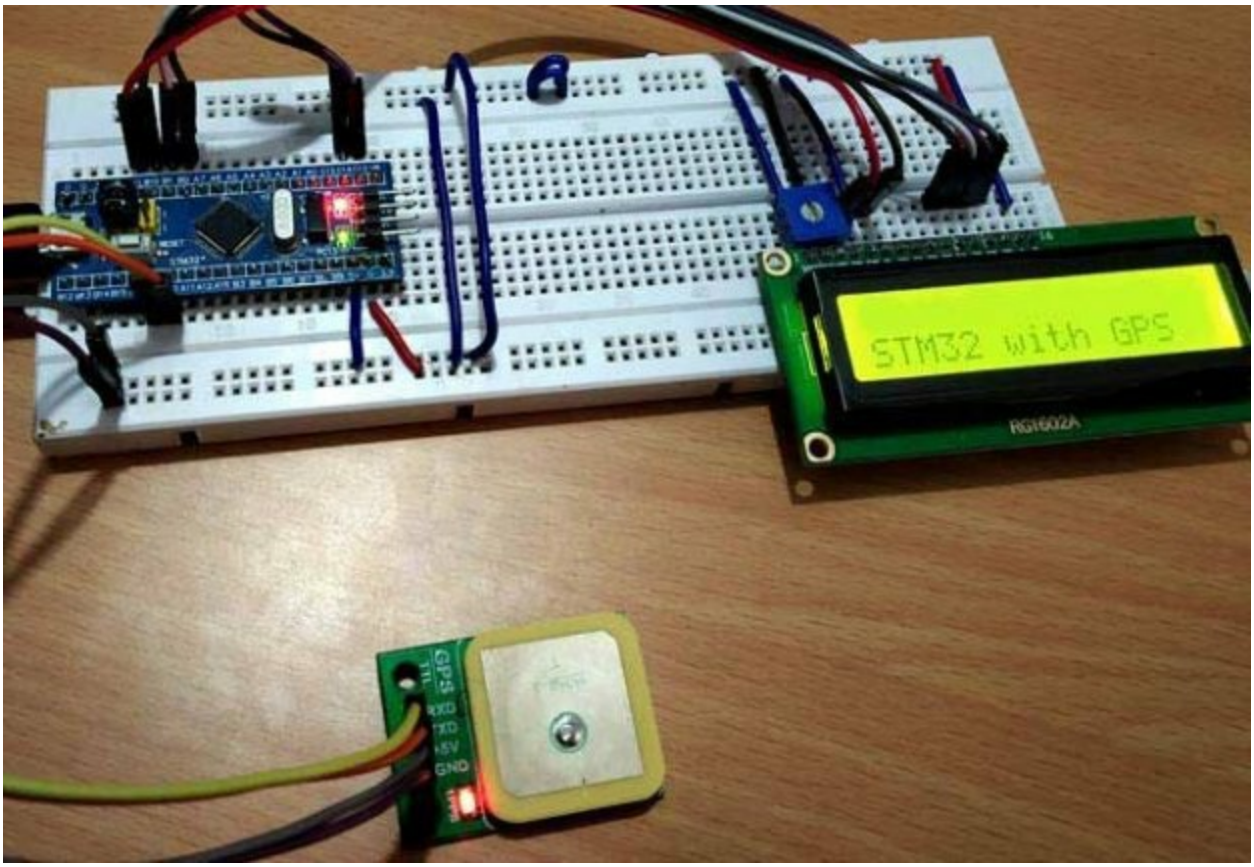
GPS Module	STM32F103C8
RXD	PA9 (TX1)
TXD	PA10 (RX1)
+5V	+5V
GND	GND

Associations between 16x2 LCD and STM32F103C8

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 entire arrangement will look like underneath:



Programming STM32F103C8 for GPS Module Interfacing

Complete program for discovering area utilizing GPS module utilizing STM32 is given toward the finish of this undertaking. STM32F103C8 can be adapted utilizing Arduino IDE by essentially associating it to PC through USB port. Try to evacuate the pins TX and RX while transferring code and interface it in the wake of transferring.

To interface GPS with STM32, 1st we require to download a library from the GitHub connect TinyGPSPlus. In the wake of downloading the library, it tends to be remembered for the Arduino IDE by to Sketch - > Include Library

- > Add .zip Library. Same library can be utilized to interface GPS with Arduino.

So first incorporate the essential library documents and characterize pins for 16x2 LCD:

```
#include <LiquidCrystal.h>

#include <TinyGPS++.h>

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

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

At that point make an item named gps of the class TinyGPSPlus.

```
TinyGPSPlus gps;
```

Next in the void arrangement, start the sequential correspondence with the GPS module utilizing Serial1.begin(9600). Serial1 is utilized as the Serial 1 port (Pins-PA9, PA10) of the STM32F103C8.

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

At that point show invite message for quite a while.

```
  lcd.begin(16,2);

  lcd.print("Hello_world");

  lcd.setCursor(0,1);
```

```
  lcd.print("STM32 with GPS");

  delay(4000);

  lcd.clear();
```

Next in the void circle (), we get scope and longitude from the GPS and check whether the information got is substantial or not and show data in the sequential screen and LCD.

Checking if the area information accessible is substantial or not

```
loc_valid = gps.location.isValid();
```

Gets the scope information

```
lat_val = gps.location.lat();
```

Gets the longitude information

```
  lng_val = gps.location.lng();
```

In case invalid information is gotten it shows "*****" in sequential screen and show "pausing" in LCD.

```
if (!loc_valid)

  {

    lcd.print("Waiting");

    Serial.print("Latitude : ");
```

```
  Serial.println("*****");
  Serial.print("Longitude : ");
  Serial.println("*****");
  delay(4000);
  lcd.clear();
}
```

In case substantial information is gotten the scope and longitude is shown on sequential screen just as on LCD show.

```
lcd.clear();
  Serial.println("GPS READING: ");
  Serial.print("Latitude : ");
  Serial.println(lat_val, 6);
  lcd.setCursor(0,0);
  lcd.print("LAT:");
  lcd.print(lat_val,6);
  Serial.print("Longitude : ");
   Serial.println(lng_val, 6);
   lcd.setCursor(0,1);
```

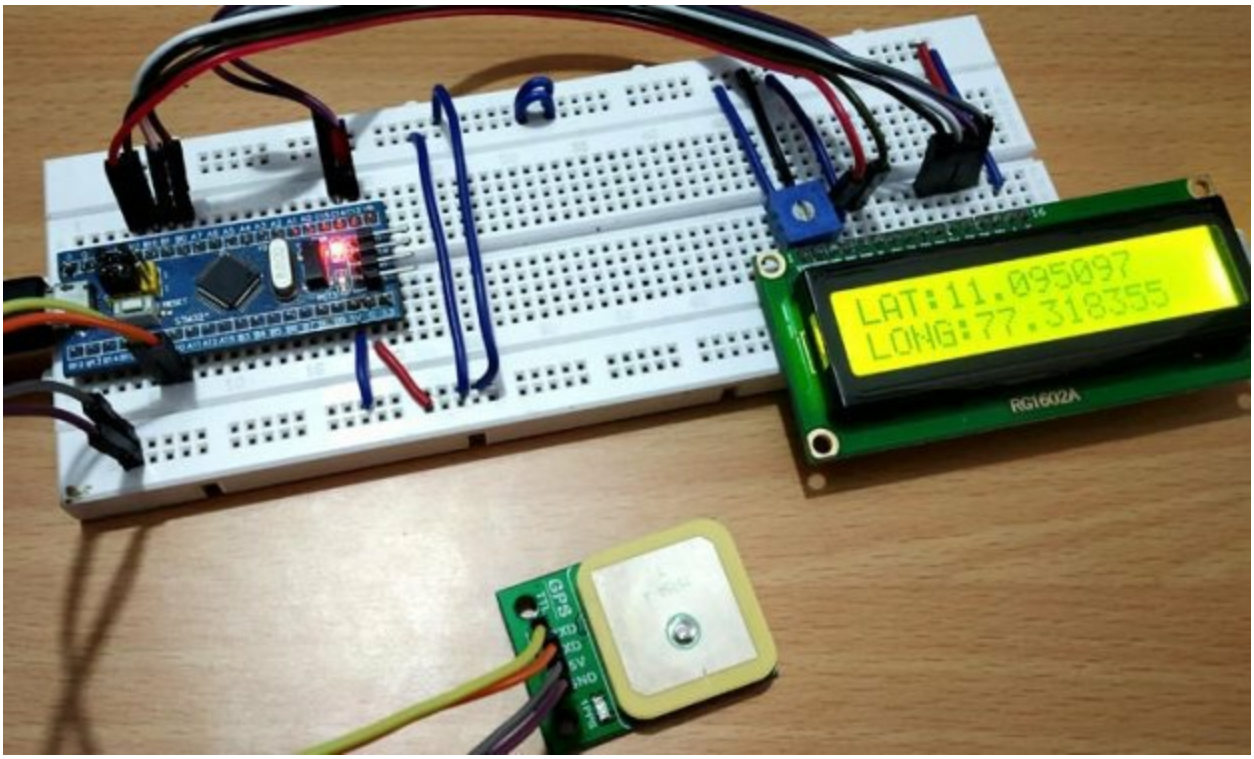
```
      lcd.print("LONG:");

      lcd.print(lng_val,6);

      delay(4000);
```

Following capacity gives the deferral to peruse the information. It continues searching for the information on sequential port.

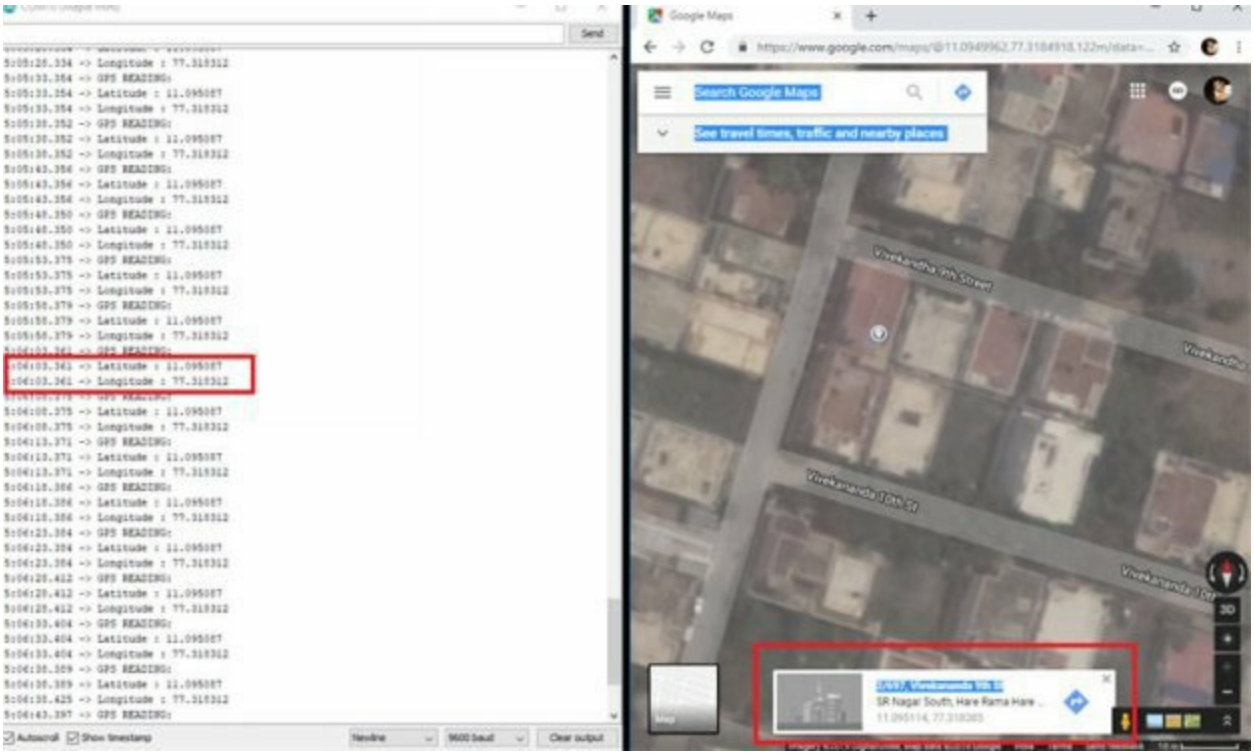
```
static void GPSDelay(unsigned long ms)

{

  unsigned long start = millis();

  do

  {

    while (Serial1.available())

    gps.encode(Serial1.read());

  } while (millis() - start < ms);

}
```

Discovering Latitude and Longitude with GPS and STM32

In the wake of building the arrangement and transferring the code, make a point to put the GPS module in open zone to get the sign quick. Now and then it takes few moments to get signal so sit tight for quite a while. The Light Emitting Diode will begin flickering in the Global Positioning System module when it begins to get sign and area directions will be shown on the Liquid Crystal Display show.



You can check the scope along with longitude of area by utilizing Google maps. Simply Go to Google maps with Global Positioning System turned ON along with click on the blue dab. It will show the location with the scope along with longitude as appeared in the image beneath



The total code is given underneath.

Code

```
#include <LiquidCrystal.h>      //Library for using LCD display functions
#include <TinyGPS++.h>          //Library for using GPS functions

const int rs = PB11, en = PB10, d4 = PB0, d5 = PB1, d6 = PC13, d7 = PC14; //LCD pins connected with STM32
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

TinyGPSPlus gps;                //Object gps for class TinyGPSPlus

void setup()
{
  Serial1.begin(9600);          //Begins Serial comunication at Serial Port 1 at 9600 baudrate
  lcd.begin(16,2);              //Sets display in 16x2 Mode
  lcd.print("Hello_world");
```

```
  lcd.setCursor(0,1);
  lcd.print("STM32 with GPS");
  delay(4000);
  lcd.clear();
}
```

```
void loop()
{
  GPSDelay(1000);
  unsigned long start;
  double lat_val, lng_val;
  bool loc_valid;
  lat_val = gps.location.lat();        //Gets the latitude
  loc_valid = gps.location.isValid();
  lng_val = gps.location.lng();        //Gets the longitude


   if (!loc_valid)                    //Excecutes when invalid data is received from
GPS
  {
    lcd.print("Waiting");
    Serial.print("Latitude : ");
    Serial.println("*****");
    Serial.print("Longitude : ");
    Serial.println("*****");
    delay(4000);
    lcd.clear();
  }
  else                               //Excutes when valid data is received from GPS
  {
    lcd.clear();


      Serial.println("GPS READING: ");


      Serial.print("Latitude : ");
    Serial.println(lat_val, 6);   //Prints latitude at Serial Monitor
```

```
    lcd.setCursor(0,0);
   lcd.print("LAT:");
   lcd.print(lat_val,6);         //Prints latitude at LCD display
```

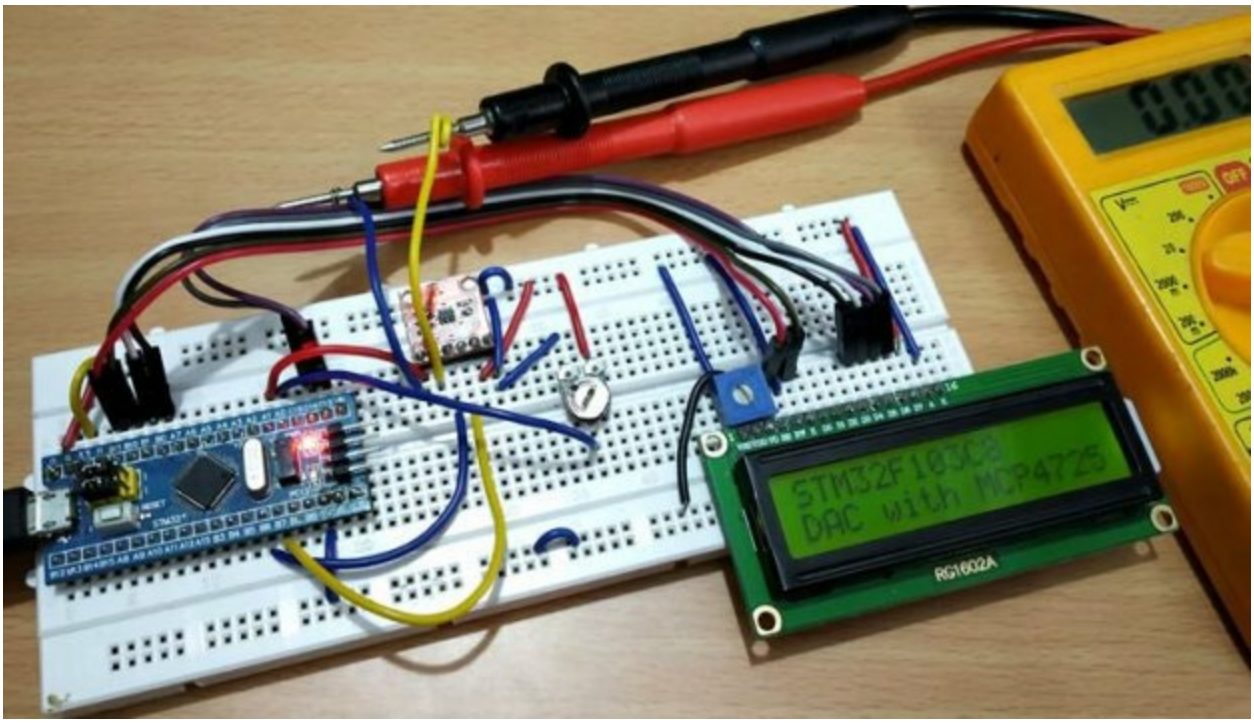
```
   Serial.print("Longitude : ");
   Serial.println(lng_val, 6);   //Prints longitude at Serial monitor
```

```
   lcd.setCursor(0,1);
   lcd.print("LONG:");
   lcd.print(lng_val,6);          //Prints longitude at LCD display
```

```
   delay(4000);
  }
}
```

```
static void GPSDelay(unsigned long ms)          //Delay for receiving data
from GPS
{
  unsigned long start = millis();
  do
  {
    while (Serial1.available())
    gps.encode(Serial1.read());
  } while (millis() - start < ms);
}
```

9. THE MOST EFFECTIVE METHOD TO UTILIZE DIGITAL-TO-ANALOG CONVERTER (DAC) WITH STM32F10C8 BOARD



We as a whole realize that the Microcontrollers work just with computerized esteems however in certifiable we need to manage simple signs. That is the reason ADC (Analog to Digital Converters) is there to change over true Analog qualities into Digital structure so microcontrollers can process the signs. Be that as it may, imagine a scenario where we need Analog signs from advanced qualities, so here comes the DAC (Digital to Analog Converter.

A basic model for Digital to Analog converter is recording a melody in studio where a craftsman artist is utilizing amplifier and singing a tune. These simple sound waves are changed over into advanced structure and afterward put away in a computerized design document and when the tune is played utilizing the put away advanced record those computerized qualities are changed over into simple signs for speaker yield. So in this framework DAC is utilized.

DAC can be utilized in numerous applications, for example, Motor control, Control Brightness of the LED Lights, Audio Amplifier, Video Encoders, Data Acquisition Systems along with etc.

We as of now interfaced MCP4725 DAC Module with Arduino. Today we

will utilize the equivalent MCP4725 DAC IC to structure a Digital to Analog converter utilizing the STM32F103C8 Microcontroller.

Parts Required

- STM32F103C8
- MCP4725 DAC IC
- 10k Potentiometer
- 16x2 LCD show
- Breadboard
- Interfacing Wires

MCP4725 DAC Module (Digital to Analog Converter)

MCP4725 IC is a 12-Bit Digital to Analog Converter Module which is utilized to create yield simple voltages from (0 to 5V) and it is constrained by utilizing I2C correspondence. It additionally accompanies on board nonvolatile memory EEPROM.

This IC has 12-Bit goals. This implies we use (0 to 4096) as contribution to furnish the voltage yield regarding reference voltage. Most extreme reference voltage is 5V.

Recipe to compute Output Voltage

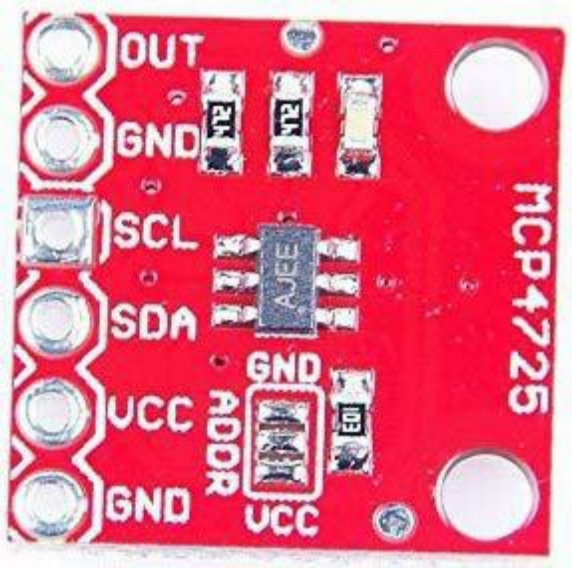
O/P Voltage = (Reference Voltage / Resolution) x Digital Value

For Example in the event that we utilize 5V as reference voltage and how about we expect that computerized esteem is 2048. So to compute the DAC yield.

O/P Voltage = (5/ 4096) x 2048 = 2.5V

Pinout of MCP4725

The following is the picture of MCP4725 with plainly showing pin names.



Pins of MCP4725	Use
OUT	Outputs Analog Voltage
GND	GND for Output
SCL	I2C Serial Clock line
SDA	I2C Serial Data line
VCC	Input Reference Voltage 5V or 3.3V

GND	GND for input

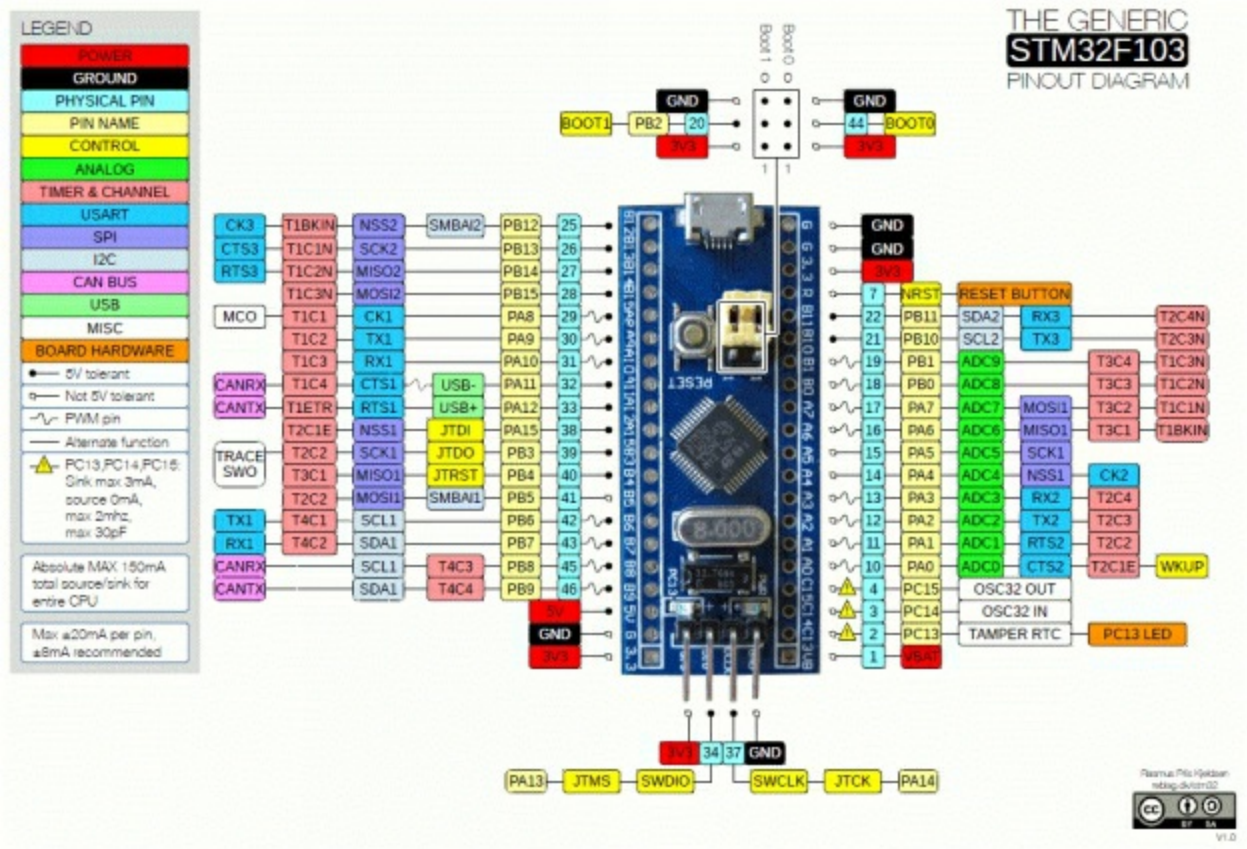
I2C Communication in MCP4725

This DAC IC can be interfaced with any microcontroller utilizing the I2C correspondence. I2C correspondence requires just two wires SCL and SDA. As a matter of course, the I2C address for MCP4725 is 0x60. Follow the connection to find out about I2C correspondence in STM32F103C8.

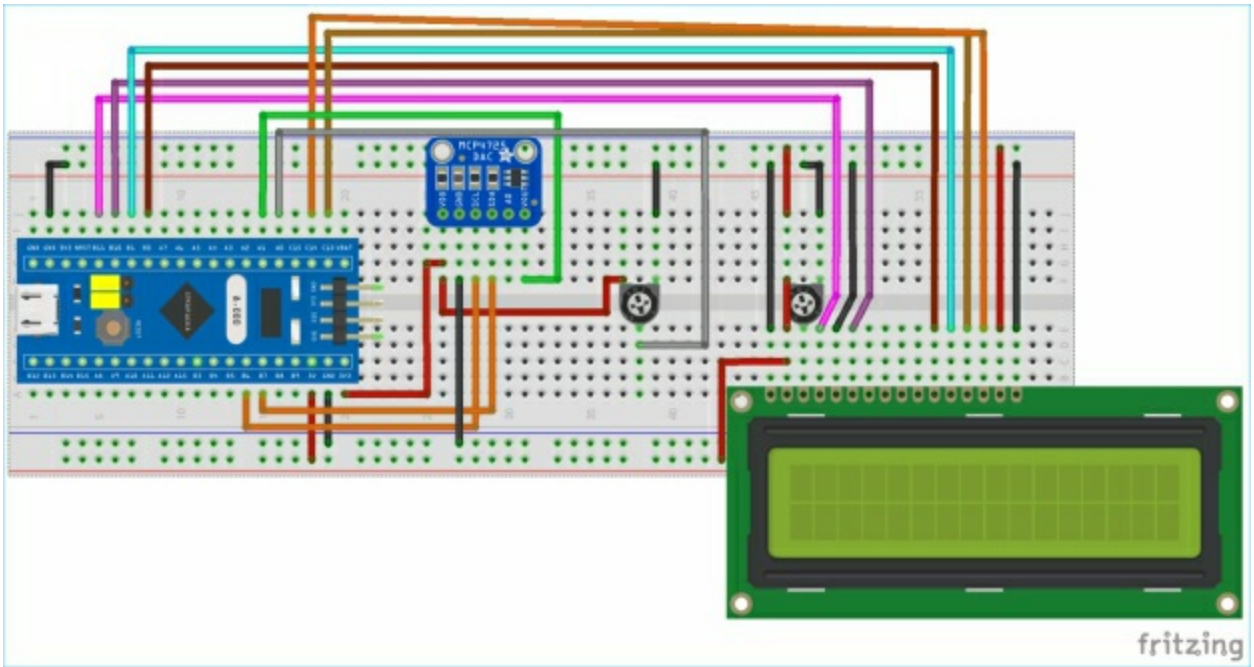
I2C sticks in STM32F103C8:

SDA: PB7 or PB9, PB11.

SCL: PB6 or PB8, PB10.



Circuit Diagram and Explanation



Associations between STM32F103C8 and 16x2 LCD

LCD Pin No	LCD Pin Name	STM32 Pin Name
1	Ground (Gnd)	Ground (G)
2	VCC	5V
3	VEE	Pin from Centre of Potentiometer for contrast
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)

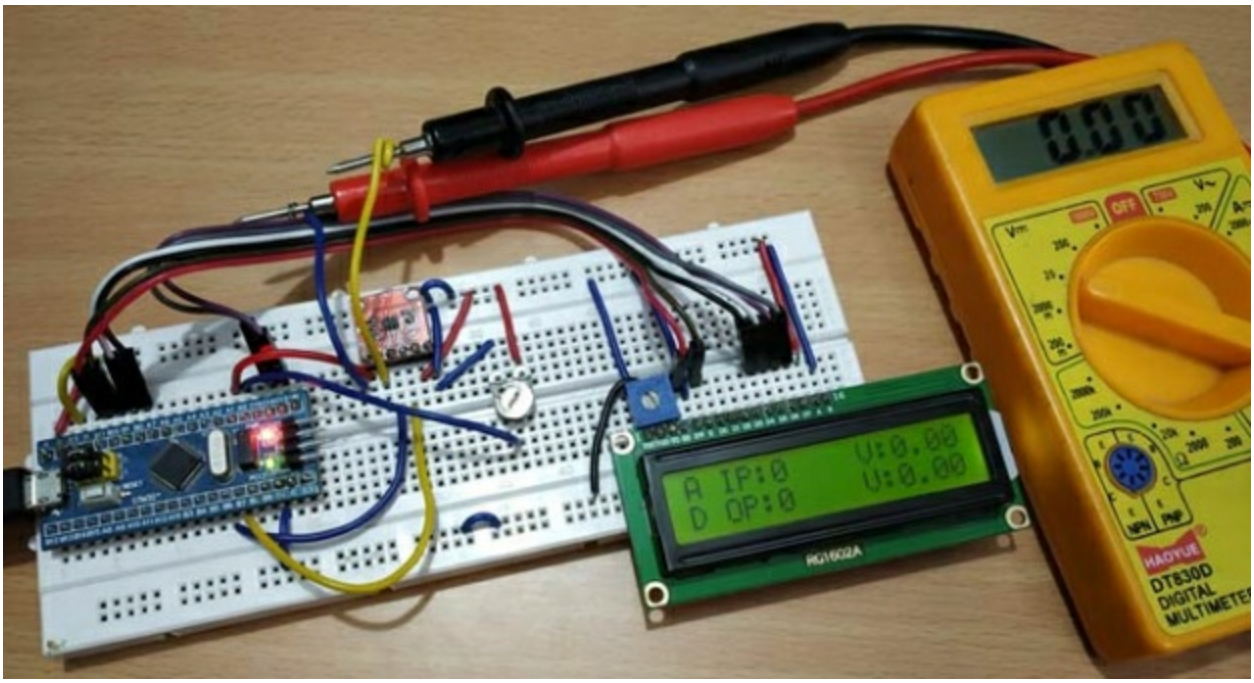
Association between MCP4725 DAC IC and STM32F103C8

MCP4725	**STM32F103C8**	**Multimeter**
SDA	PB7	NC

SCL	PB6	NC
OUT	PA1	Positive Probe
GND	GND	Negative Probe
VCC	3.3V	NC

A potentiometer is likewise associated, with focus pin associated with PA1 simple information (ADC) of STM32F10C8, Left Pin associated with GND and right most pin associated with 3.3V of STM32F103C8.

In this instructional exercise we will associate a MCP4725 DAC IC with STM32 and utilize a 10k potentiometer to give simple info incentive to STM32 ADC pin PA0. And afterward use ADC to change over simple incentive into computerized structure. After that send those computerized qualities to MCP4725 through I2C transport. At that point convert those advanced qualities to simple utilizing the DAC MCP4725 IC and afterward utilize another ADC pin PA1 of STM32 to check the simple yield of MCP4725 from the pin OUT. At last showcase the both ADC and DAC values with voltages in the 16x2 LCD show.



Programming STM32F103C8 for Digital to Analog Conversion

A FTDI software engineer isn't required currently to transfer code to STM32F103C8. Just interface it to PC by means of USB port of STM32 and begin programming with ARDUINO IDE. Visit this connect to get familiar with Programming your STM32 in Arduino IDE. Complete program for this STM32 DAC instructional exercise is given toward the end.

First incorporate library for I2C and LCD utilizing wire.h , SoftWire.h and liquidcrystal.h library. Get familiar with I2C in STM32 Microcontroller here.

```
#include<Wire.h>

#include <LiquidCrystal.h>

#include<SoftWire.h>
```

Next characterize and introduce the LCD pins as indicated by the LCD pins associated with the STM32F103C8

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

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

At that point characterize the I2C address of the MCP4725 DAC IC. The MCP4725 DAC default I2C address is 0x60

```
#define MCP4725 0x60
```

In the void arrangement()

Initially start the I2C correspondence at the pins PB7 (SDA) and PB6 (SCL) of STM32F103C8.

```
Wire.begin();     //Begins the I2C communication
```

Next set the LCD show in the 16x2 mode and show an invite message.

```
  lcd.begin(16,2);

  lcd.print("Hello_World");

  delay(1000);

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print("STM32F103C8");

  lcd.setCursor(0,1);

  lcd.print("DAC with MCP4725");
```

```
delay(2000);

lcd.clear();
```

In the void circle()

1. First in buffer[0] put the control byte esteem (0b01000000).

```
(010-Sets MCP4725 in Write mode)

buffer[0] = 0b01000000;
```

2. Following proclamation peruses the simple incentive from pin PA0 and changes over it into advanced worth running from 0 to4096 as ADC is 12-piece goals and store in the variable adc.

```
adc = analogRead(PA0) ;
```

3. This following articulation is an equation used to figure the voltage from the ADC input esteem (0 to 4096) with the reference voltage of 3.3V.

```
float ipvolt = (3.3/4096.0)* adc;
```

4. Put the Most notable piece esteems in buffer[1] by moving 4 bits to directly in ADC variable, and Least critical piece esteems in buffer[2] by moving 4 bits to left in adc variable.

```
buffer[1] = adc >> 4;

buffer[2] = adc << 4;
```

5. The accompanying explanation peruses simple incentive from ADC pin PA1 of STM32 that is the DAC yield (MCP4725 DAC IC's OUTPUT pin).

This pin can similarly be associated with multimeter to check the yield voltage.

```
unsigned int analogread = analogRead(PA1);
```

6. Further the voltage esteem from the variable analogread is determined utilizing the equation with the accompanying explanation.

```
float opvolt = (3.3/4096.0)* analogread;
```

7. In a similar void circle () there are hardly any different explanations which are clarified beneath

Starts the transmission with MCP4725:

```
Wire.beginTransmission(MCP4725);
```

Sends the control byte to I2C

```
Wire.write(buffer[0]);
```

Sends the MSB to I2C

```
Wire.write(buffer[1]);
```

Sends the LSB to I2C

```
Wire.write(buffer[2]);
```

Parts of the bargains

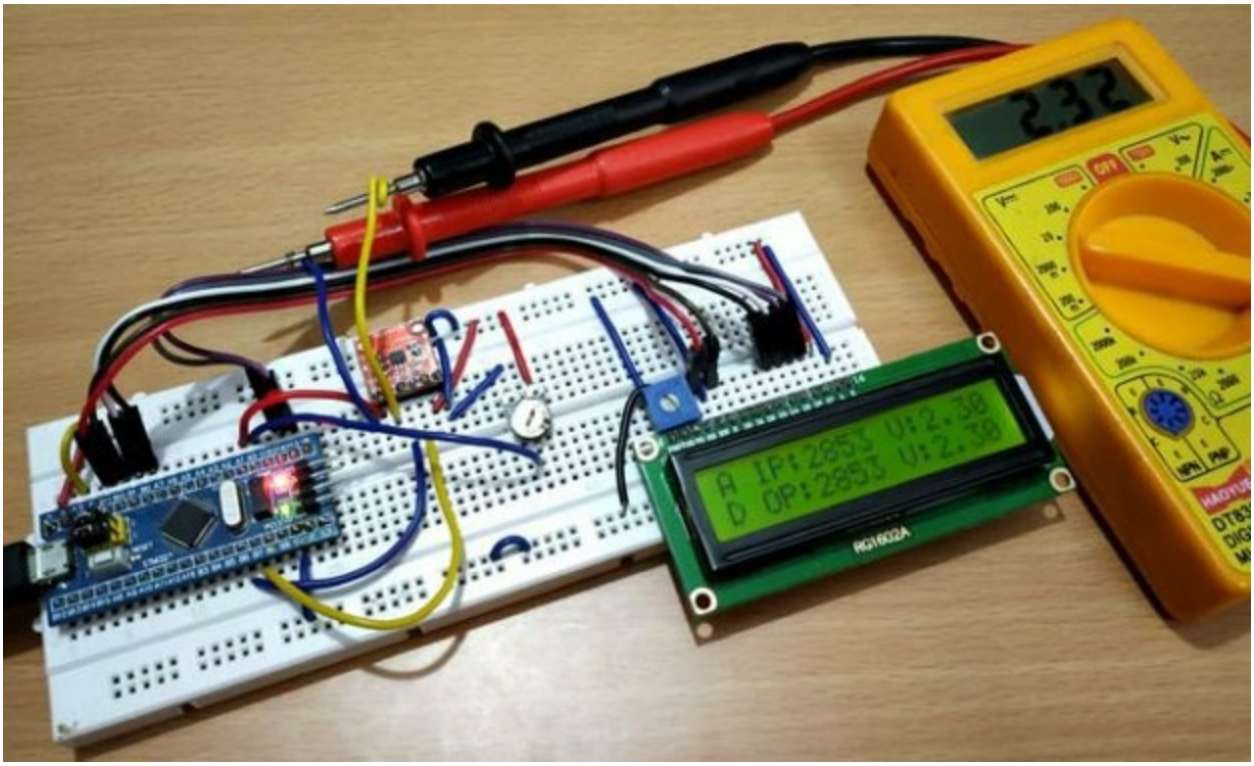
```
Wire.endTransmission();
```

Presently show those outcomes in the LCD 16x2 showcase utilizing lcd.print()

```
lcd.setCursor(0,0);
  lcd.print("A IP:");
  lcd.print(adc);
  lcd.setCursor(10,0);
  lcd.print("V:");
  lcd.print(ipvolt);
  lcd.setCursor(0,1);
  lcd.print("D OP:");
  lcd.print(analogread);
  lcd.setCursor(10,1);
  lcd.print("V:");
  lcd.print(opvolt);
  delay(500);
  lcd.clear();
```

Testing the DAC with STM32

At the point when we shift the information ADC worth and voltage by turning the potentiometer, the yield DAC worth and voltage additionally changes. Here the info esteems are appeared in the principal line and yield esteems in second line of LCD show. A multimeter is likewise associated with the MCP4725 Output Pin to confirm the simple voltage.



Complete code is given underneath.

Code

```
#include<Wire.h>                    //Include Wire library for using I2C functions
#include<SoftWire.h>
#include <LiquidCrystal.h>          //Include LCD library for using LCD display functions
```

```
#define MCP4725 0x60                //MCP4725 address as 0x60 Change yours
```

accordingly

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

```
unsigned int adc;
byte buffer[3];
```

```
void setup()
{
  Wire.begin();                  //Begins the I2C communication
  lcd.begin(16,2);                //Sets LCD in 16X2 Mode
  lcd.print("Hello_world");
  delay(1000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("STM32F103C8");
  lcd.setCursor(0,1);
  lcd.print("DAC with MCP4725");
  delay(2000);
  lcd.clear();
}
```

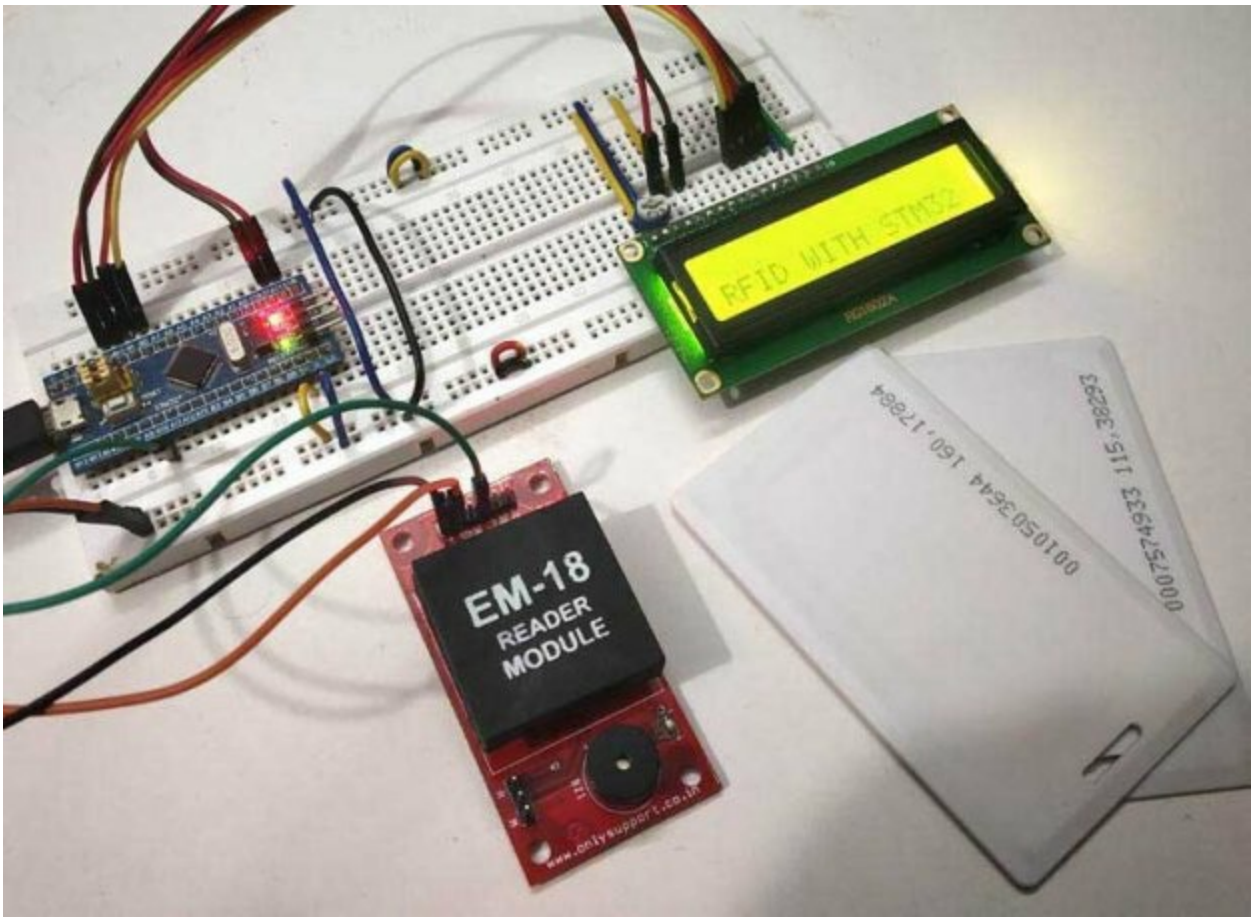
```
void loop()
```

```
{
  buffer[0] = 0b01000000;           //Sets the buffer0 with control byte (010-
Sets in Write mode)
  adc = analogRead(PA0);            //Read Analog value from pin PA0


   float ipvolt = (3.3/4096.0)* adc;  //Finding voltage formula
  buffer[1] = adc >> 4;             //Puts the most significant bit values
  buffer[2] = adc << 4;             //Puts the Least significant bit values



   unsigned int analogread = analogRead(PA1) ; //Reads analog value from
PA1
```

```
  float opvolt = (3.3/4096.0)* analogread; //Finding Voltage Formula


  Wire.beginTransmission(MCP4725);        //Joins I2C bus with MCP4725
with 0x60 address


  Wire.write(buffer[0]);           //Sends the control byte to I2C
 Wire.write(buffer[1]);           //Sends the MSB to I2C
 Wire.write(buffer[2]);           //Sends the LSB to I2C


  Wire.endTransmission();          //Ends the transmission
```

```
 lcd.setCursor(0,0);
 lcd.print("A IP:");
 lcd.print(adc);                 //Prints the ADC value from PA0
 lcd.setCursor(10,0);
 lcd.print("V:");                //Prints the Input Voltage at PA0
 lcd.print(ipvolt);
 lcd.setCursor(0,1);
 lcd.print("D OP:");
 lcd.print(analogread);           //Prints the ADC value from PA1 (From
DAC)
 lcd.setCursor(10,1);
 lcd.print("V:");
 lcd.print(opvolt);               //Prints the Input Voltage at PA1 (From DAC)


  delay(500);
 lcd.clear();
}
```

10. STEP BY STEP INSTRUCTIONS TO CONNECT RFID WITH STM32 MICROCONTROLLER



In this instructional exercise, we will structure a framework to peruse the

RFID cards utilizing STM32 along with RFID Reader. RFID represents Radio Frequency Identification which peruses data by utilizing radio frequencies. RFID's are utilized in numerous confirmation frameworks like lift stopping framework, computerized cost assortment, keeping up persistent data in medical clinics, mechanized information assortment along with.

In this instructional exercise, we will figure out how to interface an EM-18 RFID Reader Module and read the one of a kind ID of RFID labels utilizing STM32F103C8 microcontroller. We have beforehand interfaced RFID with different microcontrollers:

- RFID Interfacing with Arduino
- RFID Interfacing with PIC Microcontroller
- RFID Interfacing with MSP430 Launchpad
- RFID Interfacing with 8051 Microcontroller
- RFID along with Raspberry Pi Based Attendance System

Parts Necessary

- STM32F103C8 (Blue Pill Board)
- EM-18 RFID Reader Module
- RFID Cards
- 16x2 LCD Display Module
- Bread Board
- Interfacing Wires

Before interfacing RFID with STM32, first we will find out about RFID labels and RFID peruser.

RFID Tags

RFID labels are comprised of a microchip with a looped recieving wire that can speak with a close by peruser remotely. Distinctive sort of RFID labels with an alternate sort of shapes and sizes are accessible in the market. Not many of them utilize distinctive recurrence for correspondence reason. We will utilize 125Khz Passive RFID cards which holds the one of a kind ID information.

You can watch a curl and a microchip present inside the label when you place a RFID tag before a brilliant light.

There are fundamentally two sorts of RFID labels: Passive and Active

Aloof RFID labels draw power from the delightful field that was made by the peruser module like EM-18 and use it to control the microchip's circuits. The chip at that point sends data to the peruser.

Dynamic RFID labels requires separate force supply and contain up to 1MB of read/compose memory.

EM-18 RFID Reader

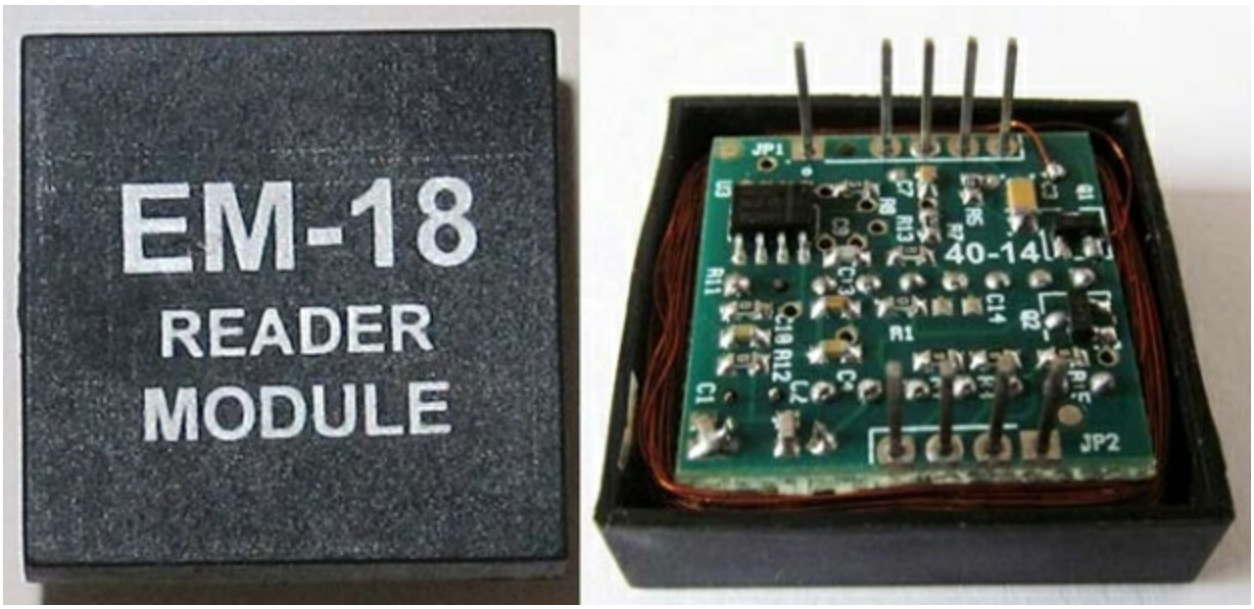
Each RFID card has an interesting ID implanted in it and a RFID peruser is utilized to peruse the RFID card no. EM-18 RFID peruser works at 125 KHz and it accompanies an on-chip reception apparatus and it very well may be controlled with 5V power supply. It gives sequential yield along weigand yield. The range is around 8-12cm. sequential correspondence parameters are

9600bps, 8 information bits, 1 stop bit. This remote RF Identification is utilized in numerous frameworks like

- RFID Based Attendance System,
- Security frameworks,
- Casting a ballot machines,
- E-cost street evaluating

Check all the RFID Projects here.

The yield gave by EM-18 RFID peruser is in 12 digit ASCII design. Out of 12 digits initial 10 digits are card number and the last two digits are the XOR aftereffect of the card number. Last two digits are utilized for blunder checking.



For instance, card number is 0200107D0D62 perused from the peruser then the card number on the card will be as beneath.

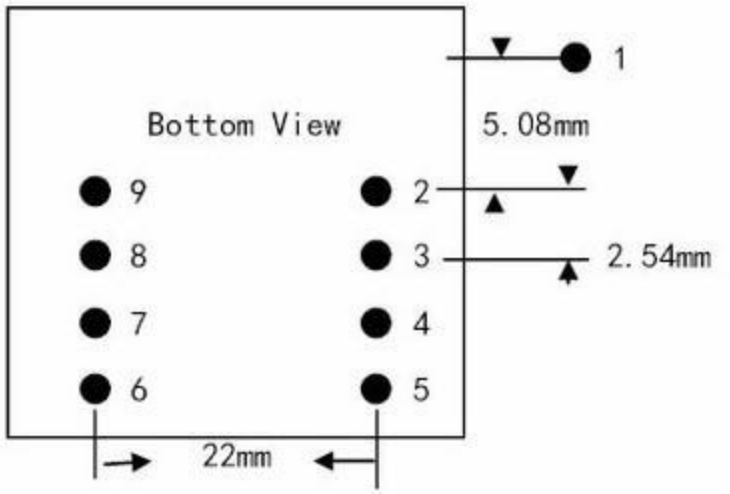
02 – preface

00107D0D = 1080589 in decimal.

62 is XOR esteem for (02 XOR 00 XOR 10 XOR 7D XOR 0D).

Thus number on the card is 0001080589.

Pin Details of EM-18



EM-18 RFID peruser has nine pins. Among nine pins, 2 pins are not associated, so we fundamentally need to think about seven terminals. The table beneath shows Pin depiction of EM-18.

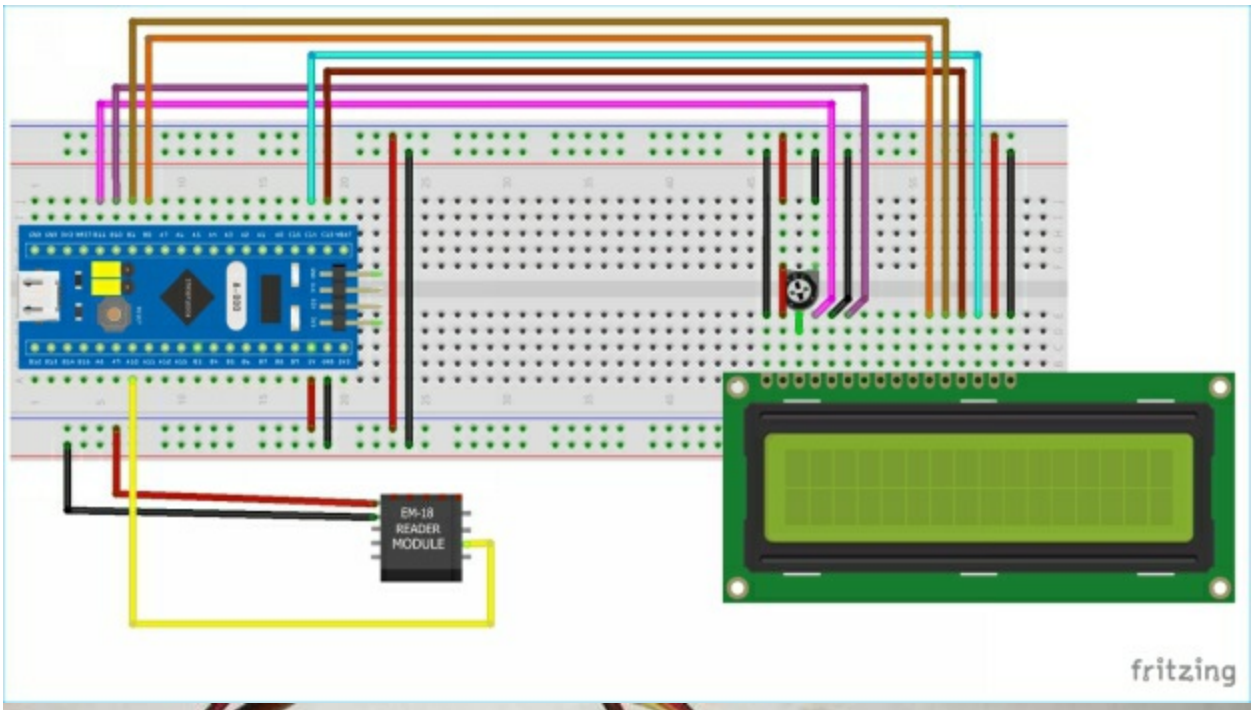
Pin Number	**Pin Name**	**Use**
1	VCC	POSITIVE
2	GND	GROUND
3	BUZZ	Connected to BUZZER
4	NC	No Connection
5	NC	No Connection
6	SEL	SEL=1 (RS232) SEL=0 (WEIGAND)
7	TX	DATA is given out through TX of RS232

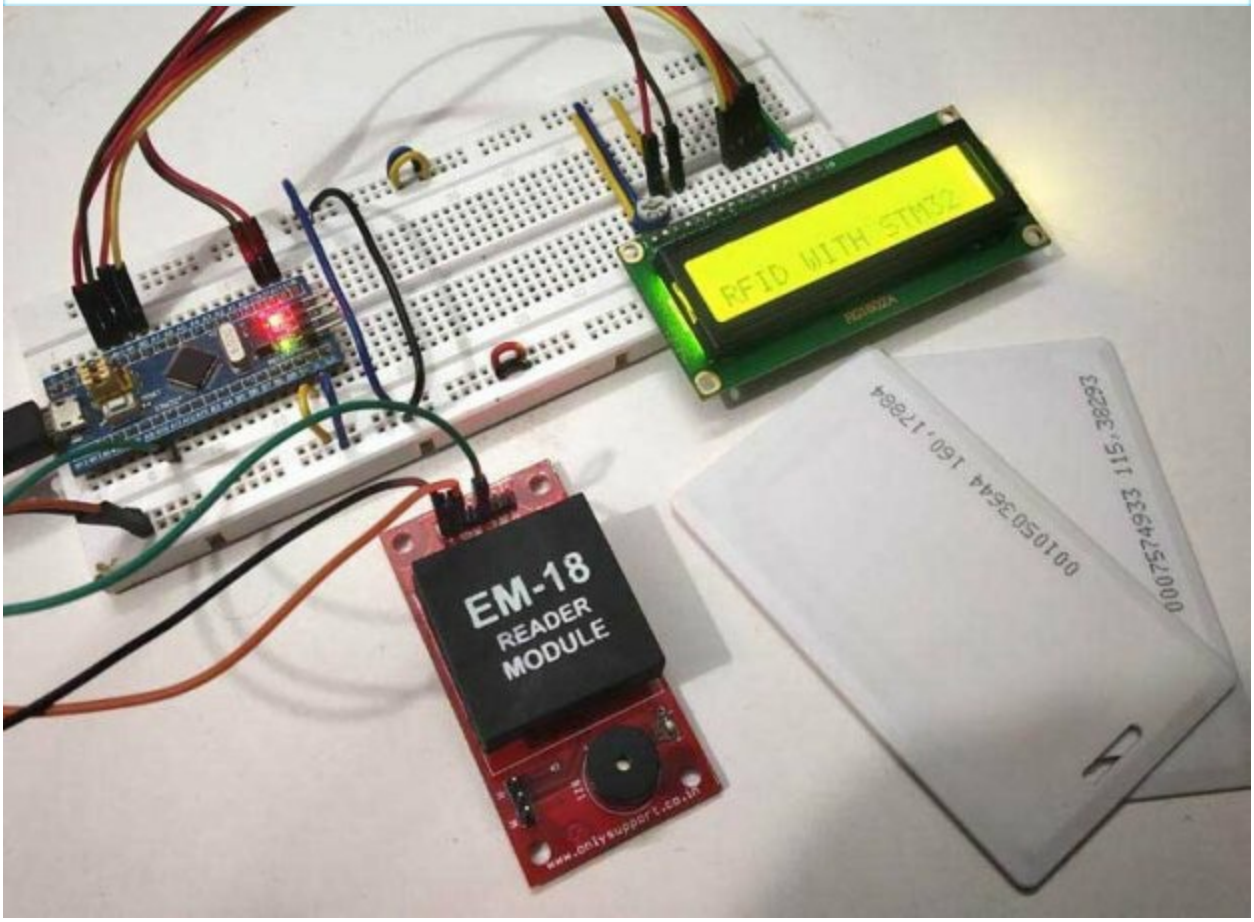
8	DATA1	WEIGAND interface DATA HIGH pin
9	DATA0	WEIGAND interface DATA LOW pin

Particulars of EM-18 RFID Module

- Understand separation: 10cm
- Working temperature: 0ºC to +80ºC
- Correspondence parameter: 9600bps
- Current Consumption: <50mA
- Working recurrence: 125 kHz
- Working Voltage: 5v

Circuit Diagram and Connections





Associations between STM32F103C8 and 16x2 LCD

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)

Associations between STM32F103C8 and EM-18 Reader Module

EM-18 Reader Module	**STM32F103C8**
VCC	+5V
GND	GND
TX	PA10

Programming STM32F103C8 for perusing RFID

In our past instructional exercise, we found out about Programming STM32F103C8T6 Board utilizing USB Port . So we needn't bother with a FTDI software engineer now. Just associate it to PC by means of USB port of STM32 and begin programming with ARDUINO IDE. Programming your STM32 in ARDUINO IDE to peruse RFID tag is exceptionally straightforward

1. To begin with, incorporate LCD show library for utilizing LCD show capacities. At that point characterize LCD sticks and introduce the LCD show. To find out about interfacing LCD with STM32F103C8 follow the connection.

```
#include <LiquidCrystal.h>

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

```
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

2. Next in void arrangement()

We have to set the LCD show mode as 16x2 and start sequential correspondence at baud rate 9600 with the pin PA10 (This is the SERIAL1 Communication port RX1 of the STM32F103C8 which is associated with the EM-18 TX pin.

```
lcd.begin(16, 2);

Serial1.begin(9600);

pinMode(PA10,INPUT);
```

3. Next showcase invite message and clear after some time.

```
lcd.print("Hello_world");       //Prints at LCD display

lcd.setCursor(0, 1);             //Set courser to second line

lcd.print("RFID WITH STM32");     //Prints at LCD display

delay(5000);                    //Delay for 5 Seconds

lcd.clear();                   //Clears LCD display

lcd.setCursor(0,0);              //Sets cursor at First Line

lcd.print("RFID TAG NO:");        //Prints at LCD display

lcd.setCursor(0,1);              //Sets cursor at Second line
```

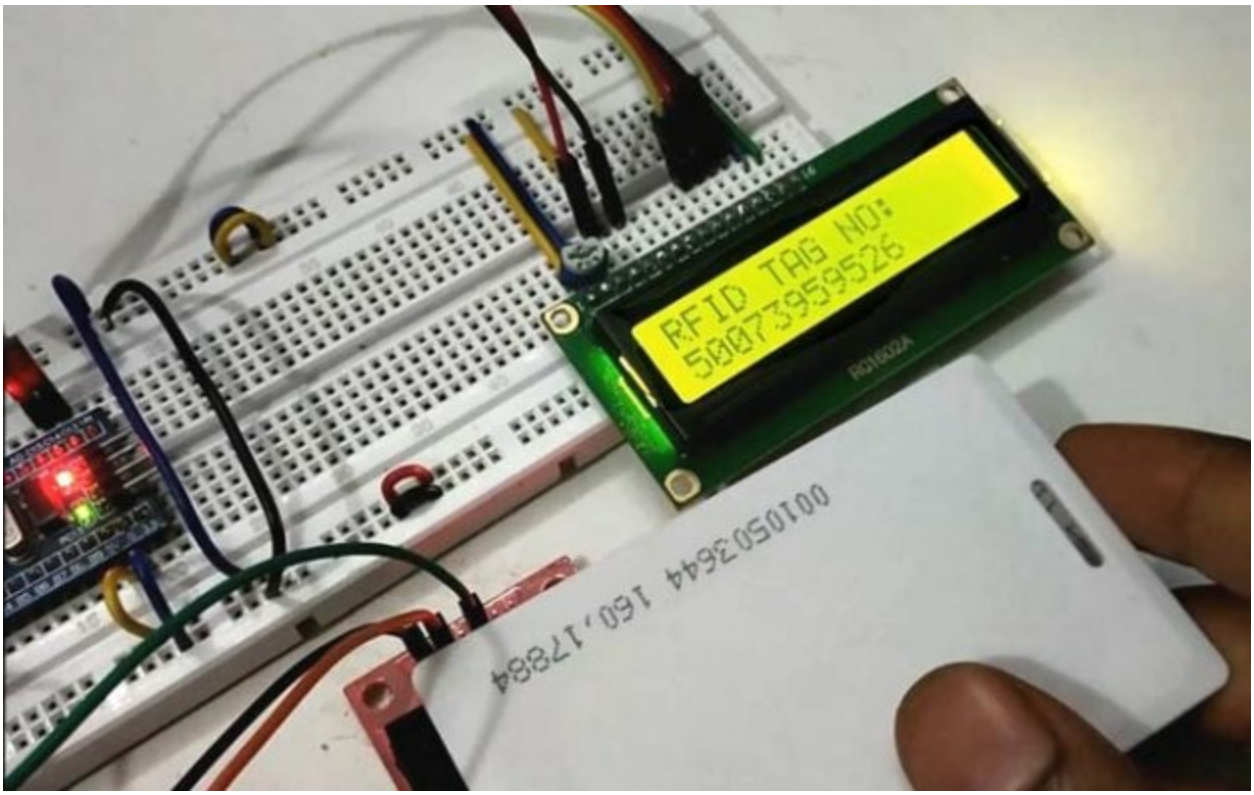
4. In void circle()

When the information from EM-18 RFID Reader Module (Tag ID) is accessible at the sequential pin of STM32F103C8 the character is put away a tiny bit at a time and showed individually on LCD show.

```
{

   while(Serial1.available() && count < 12)

  {

     RFID[count] = Serial1.read();

     count++;

     lcd.print(RFID[count]);

     if (count==12)

              {

                   lcd.print("            ");

                   count = 0;

                   lcd.setCursor(0, 1);

                }

}

}
```

Presently simply transfer the total code in STM32 and your framework is prepared to work. Simply place any RFID tag over RFID peruser and you

will see the Tag ID showing up on the 16x2 LCD show.



Complete code for utilizing RFID with STM32 microcontroller are given underneath.

Code

```
//Interfacing EM-18 RFID READER MODULE with STM32F103C8
#include <LiquidCrystal.h>                                  //Library for using LCD display functions
const int rs = PB11, en = PB10, d4 = PB0, d5 = PB1, d6 = PC13, d7 = PC14;
//mention the pin names to with LCD is connected to STM32F103C8
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);                  //Initialize the LCD display
int count = 0;
char RFID[12];                                              //Arrary for storing 12 characters of ID
```

```
void setup()
{
lcd.begin(16, 2);                // setting LCD as 16x2 type
Serial1.begin(9600);              //begins serial communication at 9600 baud
rate
pinMode(PA10,INPUT);              //Set PA10 as input pin from EM-18
lcd.print("Hello_world");       //Prints at LCD display
lcd.setCursor(0, 1);              //Set courser to second line
lcd.print("RFID WITH STM32");      //Prints at LCD display
delay(5000);                      //Delay for 5 Seconds
 lcd.clear();                     //Clears LCD display
 lcd.setCursor(0,0);              //Sets cursor at First Line
 lcd.print("RFID TAG NO:");         //Prints at LCD display
 lcd.setCursor(0,1);              //Sets cursor at Second line
}
void loop()
{
while(Serial1.available() && count < 12)          // While loop to read 12
characters and store them in input array
{
RFID[count] = Serial1.read();                //storing 12 characters one by one
   count++;
   lcd.print(RFID[count]);                    //showing 12 characters on LCD
one by one
if (count==12)
   {
lcd.print("             ");
count = 0;                        // once 12 characters are read get to start and wait
for second ID
lcd.setCursor(0, 1);              //move courser to start.
}
}
}
```

THANK YOU