

**INTERNSHIP REPORT**



**Submitted By:**

**Muhammad Asif**

**Submitted To:**

**Sir Amir Zahoor**

**Co-founder of EPTeck Technologies Pvt. Ltd.**

**EPTeck**

**4th Floor, Burj ul Kuwait Plaza, P-3B, Kohinoor City, Jaranwala Road Faisalabad,  
Punjab, Pakistan**

### **About:**

EPTeck works with the best clients worldwide to deliver the most accurate solutions for digital business problems in Embedded & IoT Services. From innovative startups with ambitious projects to digitizing worldwide corporates, we are serving digital needs with the best partners in the industry.

EPTeck is achieving international recognition with award-winning embedded products, which are attracting motivated young talent and experienced senior developers alike. Our certified business processes enable high quality development projects with a clear and structured professional outcome for our international clients.

### **ACKNOWLEDGEMENT:**

Firstly, we would like to thank ALLAH Almighty who is the creator and master of this Universe and everything in it. After that we would like to thank EPTeck especially Sir Amir Zahoor for this learning and developmental opportunity in our career.

I am highly indebted to Mr. Shahzad for their guidance and constant supervision as well as for providing necessary information regarding the internship and also for giving us his immense support.

I would also like to thank Mr. Waqas for teaching us Embedded Systems and Mr. Haider for PCB Designing.

### **PREFACE:**

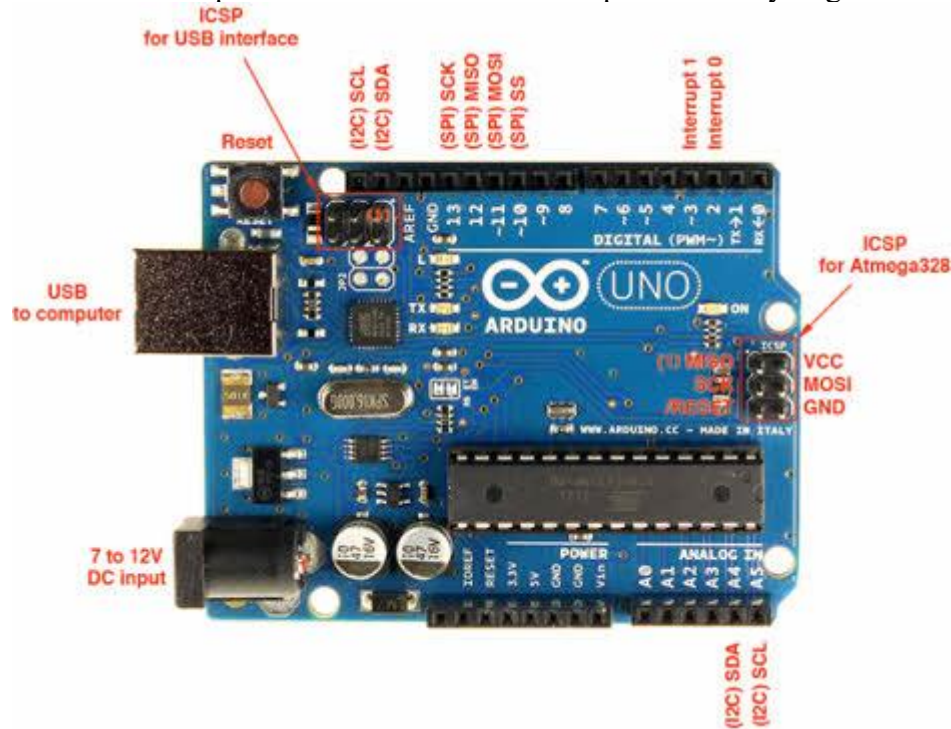
In this report, I explain each technique and method used in the company I visited during my 15-week internship at EPTeck. This report is divided into six sections. In the first element, we provided records pertaining to the SPI protocol. I2C was covered in the second section. The third component is concerned with PCB design. The fourth section is all about the STM32. The fifth part is about EPS32, and the sixth part is about networking and client- server communication.

When I joined EPTeck, communication protocols were being studied. I started studying Serial Peripheral Interface in the first month. We used MPU9250 sensor as a slave and Arduino UNO as Master to study this SPI interface thoroughly. In second month, we studied PCB Designing and STM32.

## Arduino UNO:

The Arduino Uno is a microcontroller board based on the ATmega328.

It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.



## Features of Arduino Uno Board

The features of Arduino Uno ATmega328 includes the following.

- The operating voltage is 5V
- The recommended input voltage will range from 7v to 12V
- The input voltage ranges from 6v to 20V
- Digital input/output pins are 14
- Analog i/p pins are 6
- DC Current for each input/output pin is 40 mA
- DC Current for 3.3V Pin is 50 mA
- Flash Memory is 32 KB
- SRAM is 2 KB
- EEPROM is 1 KB
- CLK Speed is 16 MHz

**Special Pins:**

Serial / UART: pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL serial chip.

External interrupts: pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

PWM (pulse-width modulation): pins 3, 5, 6, 9, 10, and 11. Can provide 8-bit PWM output with the `analogWrite()` function.

SPI (Serial Peripheral Interface): pins 10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK). These pins support SPI communication using the SPI library.

TWI (two-wire interface) / I<sup>2</sup>C: pin SDA (A4) and pin SCL (A5). Support TWI communication using the Wire library.

AREF (analog reference): Reference voltage for the analog inputs.

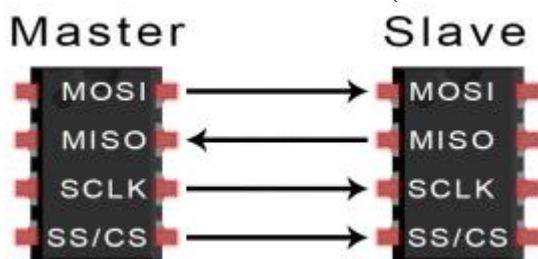
## **SPI:**

### **INTRODUCTION TO SPI COMMUNICATION:**

SPI is a common communication protocol used by many different devices. For example, SD card reader modules, RFID card reader modules, and 2.4 GHz wireless transmitter/receivers all use SPI to communicate with microcontrollers.

One unique benefit of SPI is the fact that data can be transferred without interruption. Any number of bits can be sent or received in a continuous stream. With I2C and UART, data is sent in packets, limited to a specific number of bits. Start and stop conditions define the beginning and end of each packet, so the data is interrupted during transmission.

Devices communicating via SPI are in a master-slave relationship. The master is the controlling device (usually a microcontroller), while the slave (usually a sensor, display, or memory chip) takes instruction from the master. The simplest configuration of SPI is a single master, single slave system, but one master can control more than one slave (more on this below).



### **The SPI bus specifies four logic signals:**

MOSI (Master Output/Slave Input) – Line for the master to send data to the slave.

MISO (Master Input/Slave Output) – Line for the slave to send data to the master.

SCLK (Clock) – Line for the clock signal.

SS/CS (Slave Select/Chip Select) – Line for the master to select which slave to send data to.

\*In practice, the number of slaves is limited by the load capacitance of the system, which reduces the ability of the master to accurately switch between voltage levels.

## **HOW SPI WORKS**

### **THE CLOCK:**

The clock signal synchronizes the output of data bits from the master to the sampling of bits by the slave. One bit of data is transferred in each clock cycle, so the speed of data transfer is determined by the frequency of the clock signal. SPI communication is always initiated by the master since the master configures and generates the clock signal.

Any communication protocol where devices share a clock signal is known as synchronous. SPI is a synchronous communication protocol. There are also asynchronous methods that don't use a clock signal. For example, in UART communication, both sides are set to a pre-configured baud rate that dictates the speed and timing of data transmission.

The clock signal in SPI can be modified using the properties of clock polarity and clock phase. These two properties work together to define when the bits are output and when they are sampled. Clock polarity can be set by the master to allow for bits to be output and sampled on either the rising or falling edge of the clock cycle. Clock phase can be set for output and sampling to occur on either the first edge or second edge of the clock cycle, regardless of whether it is rising or falling.

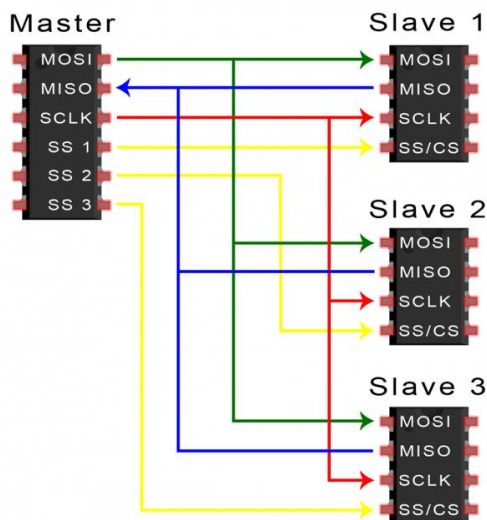
### **SLAVE SELECT:**

The master can choose which slave it wants to talk to by setting the slave's CS/SS line to a low voltage level. In the idle, non-transmitting state, the slave select line is kept at a high voltage level. Multiple CS/SS pins may be available on the master, which allows for multiple slaves to be wired in parallel. If only one CS/SS

pin is present, multiple slaves can be wired to the master by daisy-chaining.

### **MULTIPLE SLAVES SPI:**

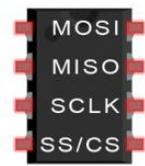
can be set up to operate with a single master and a single slave, and it can be set up with multiple slaves controlled by a single master. There are two ways to connect multiple slaves to the master. If the master has multiple slave select pins, the slaves can be wired in parallel like this:



### **MOSI AND MISO:**

The master sends data to the slave bit by bit, in serial through the MOSI line. The slave receives the data sent from the master at the MOSI pin. Data sent from the master to the slave is usually sent with the most significant bit first. The slave can also send data back to the master through the MISO line in serial. The data sent from the slave back to the master is usually sent with the least significant bit first. STEPS OF SPI DATA TRANSMISSION 1. The master outputs the clock signal:

Master



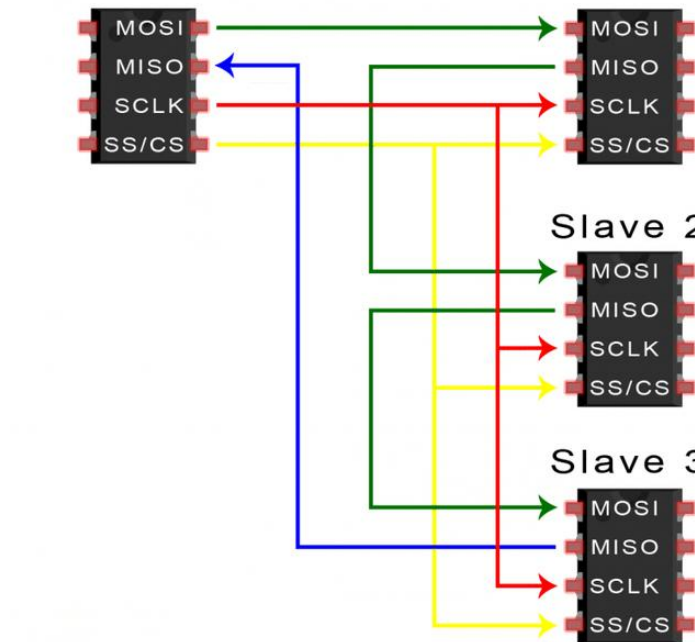
Slave 1

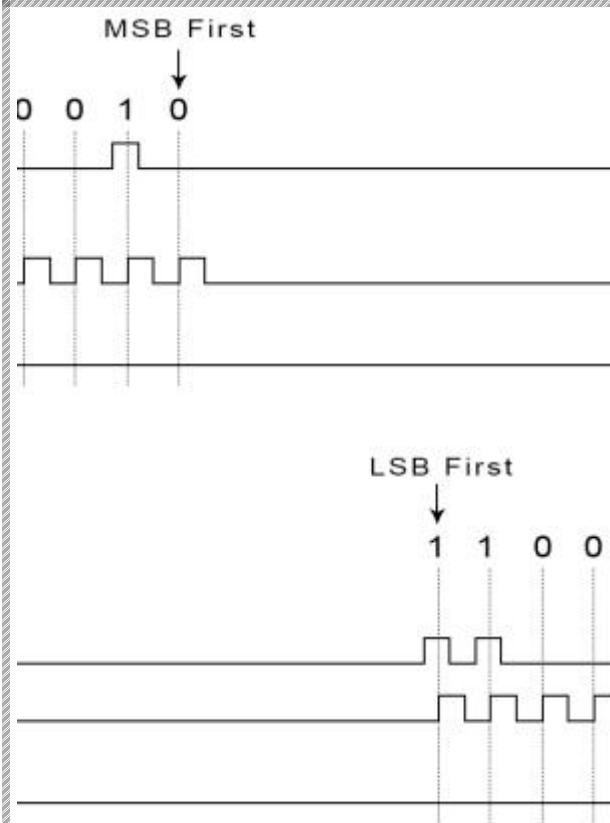


Slave 2



Slave 3





### **MOSI AND MISO:**

The master sends data to the slave bit by bit, in serial through the MOSI line. The slave receives the data sent from the master at the MOSI pin. Data sent from the master to the slave is usually sent with the most significant bit first.

The slave can also send data back to the master through the MISO line in serial. The data sent from the slave back to the master is usually sent with the least significant bit first.

### **STEPS OF SPI DATA TRANSMISSION**

**\*Note Images are Listed Oboe you can easily distinguish**

1. The master outputs the clock signal:
2. The master switches the SS/CS pin to a low voltage state, which activates the slave:
3. The master sends the data one bit at a time to the slave along the MOSI line. The slave reads the bits as they are received:
4. If a response is needed, the slave returns data one bit at a time to the master along the MISO line. The master reads the bits as they are received:



## **ADVANTAGES AND DISADVANTAGES OF SPI**

There are some advantages and disadvantages to using SPI, and if given the choice between different communication protocols, you should know when to use SPI according to the requirements of your project:

### **ADVANTAGES**

No start and stop bits, so the data can be streamed continuously without interruption No complicated slave addressing system like I2C Higher data transfer rate than I2C (almost twice as fast) Separate MISO and MOSI lines, so data can be sent and received at the same time

### **DISADVANTAGES**

Uses four wires (I2C and UARTs use two) No acknowledgement that the data has been successfully received (I2C has this) No form of error checking like the parity bit in UART Only allows for a single master Hopefully this article has given you a better understanding of SPI. Continue on to part two of this series to learn about UART driven communication, or to part three where we discuss the I2C protocol.

## I2C:

### Introduction to I2C Communication

I2C communication is the short form for inter-integrated circuits. It is a communication protocol developed by Philips Semiconductors for the transfer of data between a central processor and multiple ICs on the same circuit board using just two common wires.

[adsense1]

Owing to its simplicity, it is widely adopted for communication between microcontrollers and sensor arrays, displays, IoT devices, EEPROMs etc.

This is a type of synchronous serial communication protocol. It means that data bits are transferred one by one at regular intervals of time set by a reference clock line.

### Features

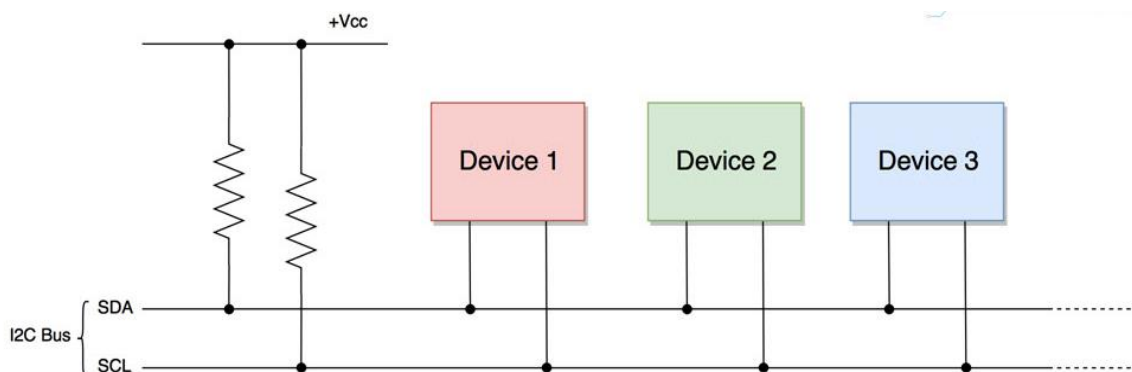
The following are some of the important features of I2C communication protocol:

- Only two common bus lines (wires) are required to control any device/IC on the I2C network
- No need of prior agreement on data transfer rate like in UART communication. So the data transfer speed can be adjusted whenever required
- Simple mechanism for validation of data transferred
- Uses 7-bit addressing system to target a specific device/IC on the I2C bus
- I2C networks are easy to scale. New devices can simply be connected to the two common I2C bus lines

### Hardware

#### The physical I2C Bus

I2C Bus (Interface wires) consists of just two wires and are named as Serial Clock Line (SCL) and Serial Data Line (SDA). The data to be transferred is sent through the SDA wire and is synchronized with the clock signal from SCL. All the devices/ICs on the I2C network are connected to the same SCL and SDA lines as shown below:



Both the I2C bus lines (SDA, SCL) are operated as open drain drivers. It means that any device/IC on the I2C network can drive SDA and SCL low, but they cannot drive them high. So, a pull up resistor is used for each bus line, to keep them high (at positive voltage) by default.

The reason for using an open-drain system is that there will be no chances of shorting, which might happen when one device tries to pull the line high and some other device tries to pull the line low.

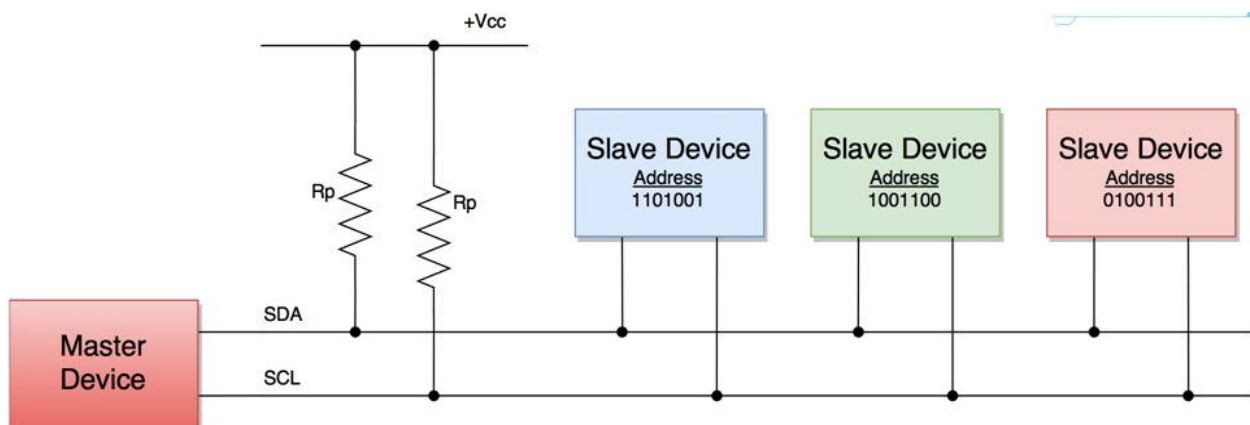
## Master and Slave Devices

The devices connected to the I2C bus are categorized as either masters or slaves. At any instant of time only a single master stays active on the I2C bus. It controls the SCL clock line and decides what operation is to be done on the SDA data line.

All the devices that respond to instructions from this master device are slaves. For differentiating between multiple slave devices connected to the same I2C bus, each slave device is physically assigned a permanent 7-bit address.

When a master device wants to transfer data to or from a slave device, it specifies this particular slave device address on the SDA line and then proceeds with the transfer. So effectively communication takes place between the master device and a particular slave device.

All the other slave devices doesn't respond unless their address is specified by the master device on the SDA line.

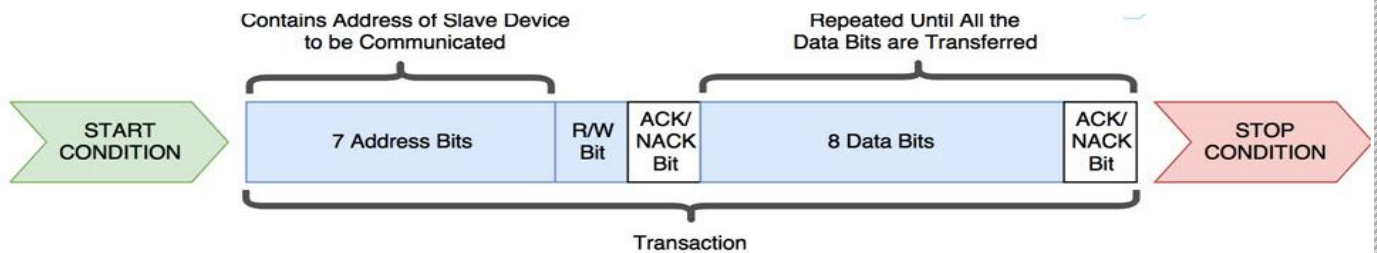


## Data Transfer Protocol

The following protocol (set of rules) is followed by master device and slave devices for the transfer of data between them.

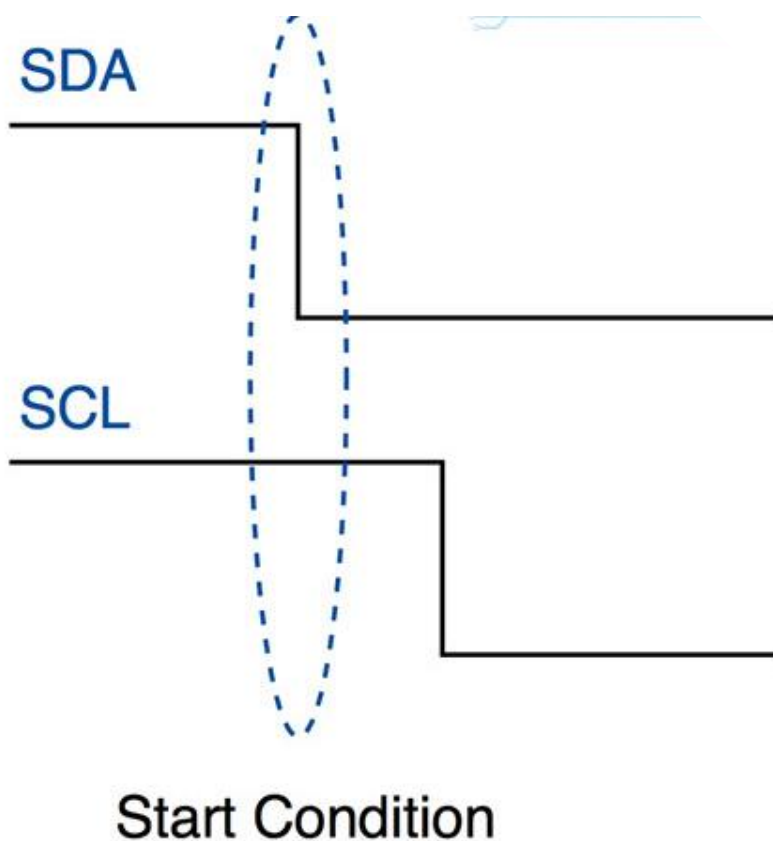
Data is transferred between the master device and slave devices through a single SDA data line, via patterned sequences of 0's and 1's (bits). Each sequence of 0's and 1's is termed as a transaction and the data in each transaction is structured as below:

## Start Condition



Whenever a master device/IC decides to start a transaction, it switches the SDA line from high voltage level to a low voltage level before the SCL line switches from high to low.

Once a start condition is sent by the master device, all the slave devices get active even if they are in sleep mode, and wait for the address bits.



## Address Block

It comprises of 7 bits and are filled with the address of slave device to/from which the master device needs send/receive data. All the slave devices on the I2C bus compare these address bits with their address.

## Read/Write Bit

This bit specifies the direction of data transfer. If the master device/IC need to send data to a slave device, this bit is set to '0'. If the master IC needs to receive data from the slave device, it is set to '1'.

### ACK/NACK Bit

It stands for Acknowledged/Not-Acknowledged bit. If the physical address of any slave device coincides with the address broadcasted by the master device, the value of this bit is set to '0' by the slave device. Otherwise it remains at logic '1' (default).

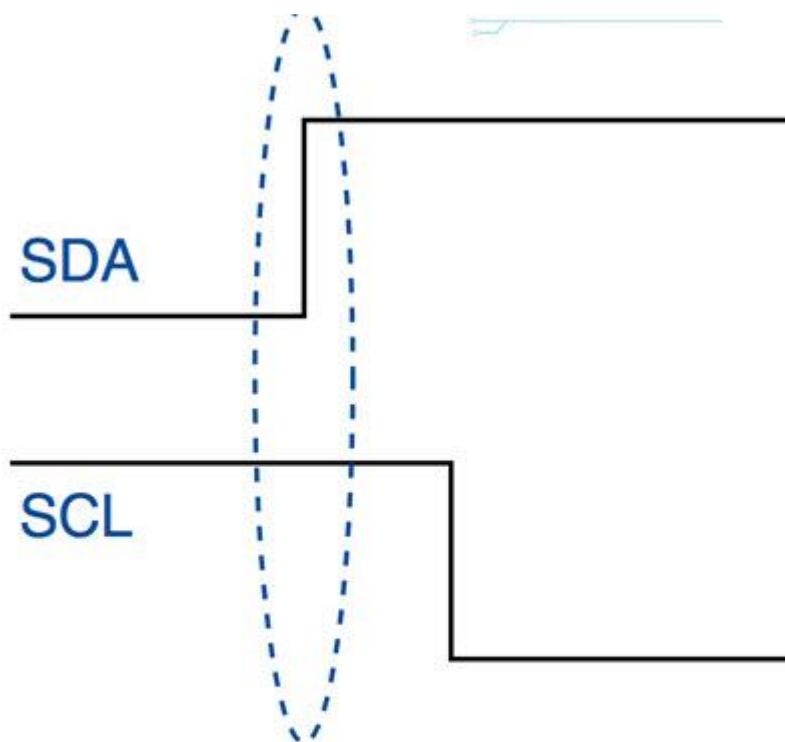
### Data Block

It comprises of 8 bits and they are set by the sender, with the data bits it needs to transfer to the receiver. This block is followed by an ACK/NACK bit and is set to '0' by the receiver if it successfully receives data. Otherwise it stays at logic '1'.

This combination of data block followed by ACK/NACK bit is repeated until the data is completely transferred.

### Stop Condition

After required data blocks are transferred through the SDA line, the master device switches the SDA line from low voltage level to high voltage level before the SCL line switches from high to low.



### Stop Condition

**NOTE:** Logic '0' or setting a bit to '0' is equivalent to applying low voltage on the SDA line and vice versa.

## How I2C Communication Practically Works?

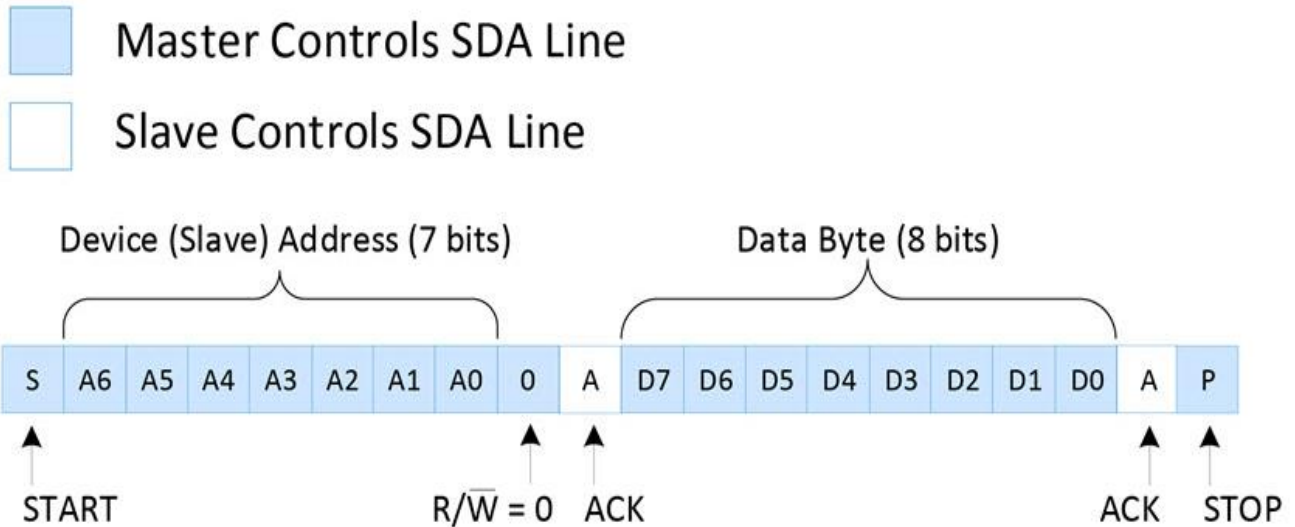
An I2C communication/transaction is initiated by a master device either to send data to a slave device or to receive data from it. Let us learn about working of both the scenarios in detail.

### Sending Data to a Slave Device

The following sequence of operations take place when a master device tries to send data to a particular slave device through I2C bus:

- The master device sends the start condition
- The master device sends the 7 address bits which corresponds to the slave device to be targeted
- The master device sets the Read/Write bit to '0', which signifies a write
- Now two scenarios are possible:
  - If no slave device matches with the address sent by the master device, the next ACK/NACK bit stays at '1' (default). This signals the master device that the slave device identification is unsuccessful. The master clock will end the current transaction by sending a Stop condition or a new Start condition
  - If a slave device exists with the same address as the one specified by the master device, the slave device sets the ACK/NACK bit to '0', which signals the master device that a slave device is successfully targeted
- If a slave device is successfully targeted, the master device now sends 8 bits of data which is only considered and received by the targeted slave device. This data means nothing to the remaining slave devices
- If the data is successfully received by the slave device, it sets the ACK/NACK bit to '0', which signals the master device to continue
- The previous two steps are repeated until all the data is transferred
- After all the data is sent to the slave device, the master device sends the Stop condition which signals all the slave devices that the current transaction has ended

The below figure represents the overall data bits sent on the SDA line and the device that controls each of them:



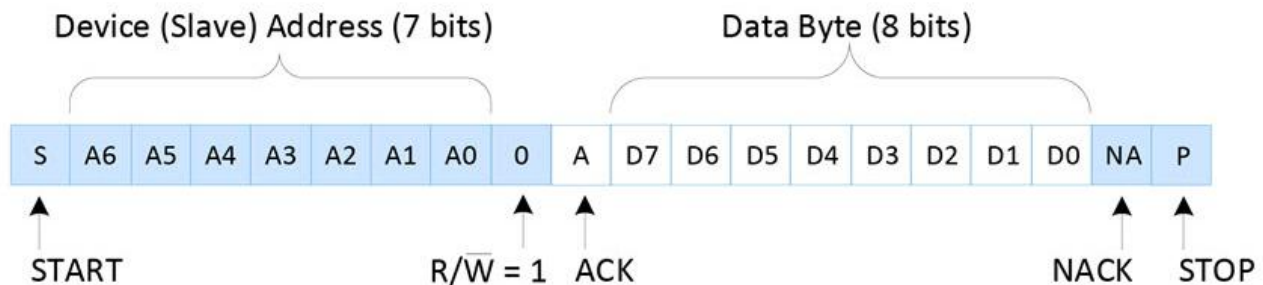
### Reading Data from a Slave Device

The sequence of operations remain the same as in previous scenario except for the following:

- The master device sets the Read/Write bit to '1' instead of '0' which signals the targeted slave device that the master device is expecting data from it
- The 8 bits corresponding to the data block are sent by the slave device and the ACK/NACK bit is set by the master device
- Once the required data is received by the master device, it sends a NACK bit. Then the slave device stops sending data and releases the SDA line

If the master device to read data from specific internal location of a slave device, it first sends the location data to the slave device using the steps in previous scenario. It then starts the process of reading data with a repeated start condition.

The below figure represents the overall data bits sent on the SDA line and the device that controls each of them:



### Concept of clock stretching

Let say the master device started a transaction and sent address bits of a particular slave device followed by a Read bit of '1'. The specific slave device needs to send an ACK bit, immediately followed by data.

But if the slave device needs some time to fetch and send data to master device, during this gap, the master device will think that the slave device is sending some data.

To prevent this, the slave device holds the SCL clock line low until it is ready to transfer data bits. By doing this, the slave device signals the master device to wait for data bits until the clock line is released.

That's it for this article. Take a look at the below projects to get an idea on how I2C communication is practically implemented, between microcontrollers/Arduinos and different sensors:

RFID Based Car Parking System

Interfacing Arduino With a Motion Processor Unit



## STM-32:

### What is STM32?

STM32 is a series of 32-bit Microcontrollers developed and marketed by the company STMicroelectronics. There are various types and varieties of STM32 Microcontrollers available, and they belong to the ARM-architecture family of Microcontrollers. These microcontrollers are used in a variety of applications, from simple printers to complex circuit boards in vehicles. As a result, the technical know-how of developing firmware engineer in electronics and communications.

### Development Tools

Development tools are required to develop the code, program the microcontroller and test/debug the code. The development tools include:

Compiler

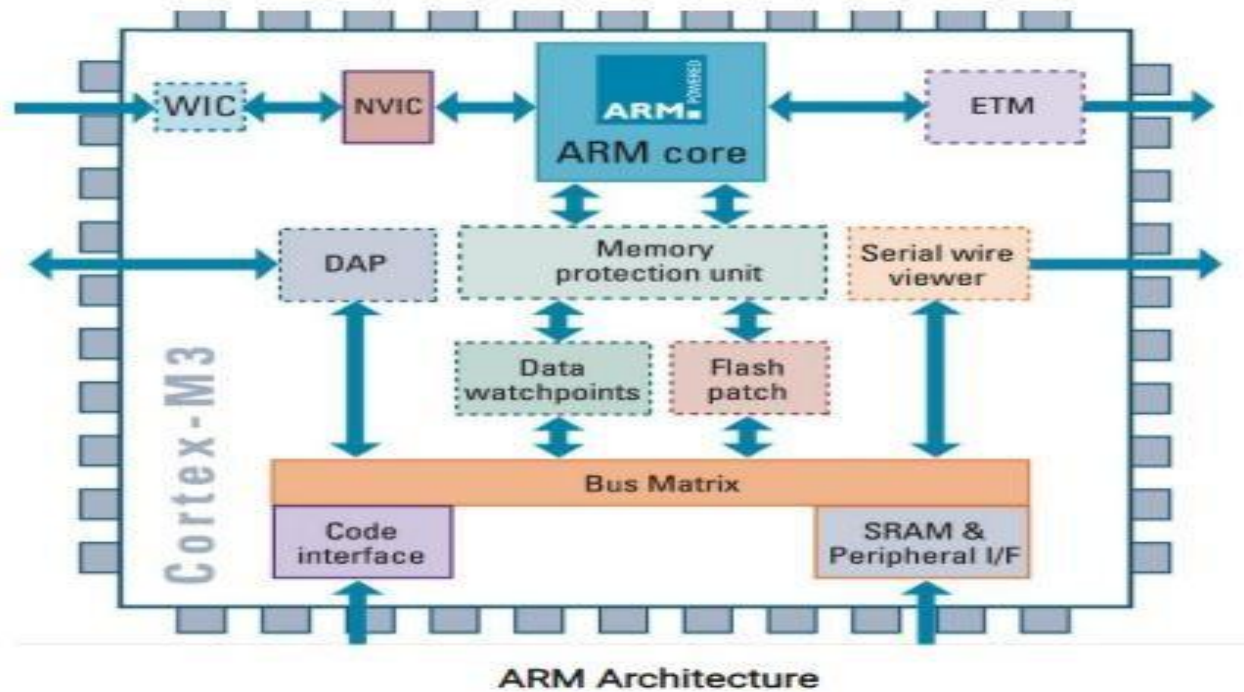
Debugger

In-Circuit Serial Programmer (ICSP)



### ARM Architecture:

ARM stands for Advanced Risk Machine. It is one of the most popular architectures used in devices like cameras, mobile phones and embedded system devices. These are known for their low-power-consumption vs better performance abilities.

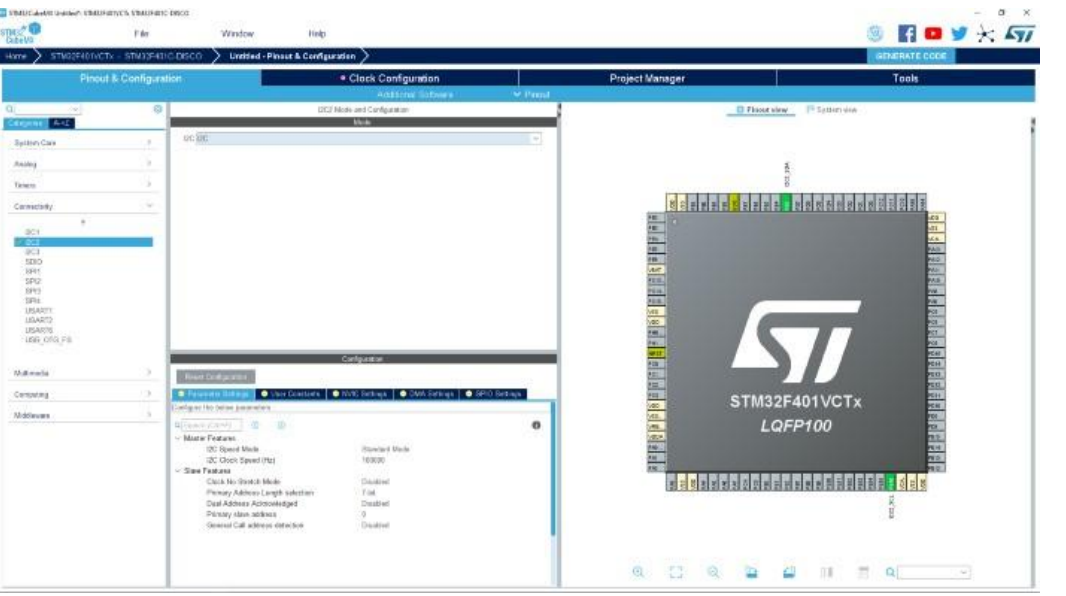
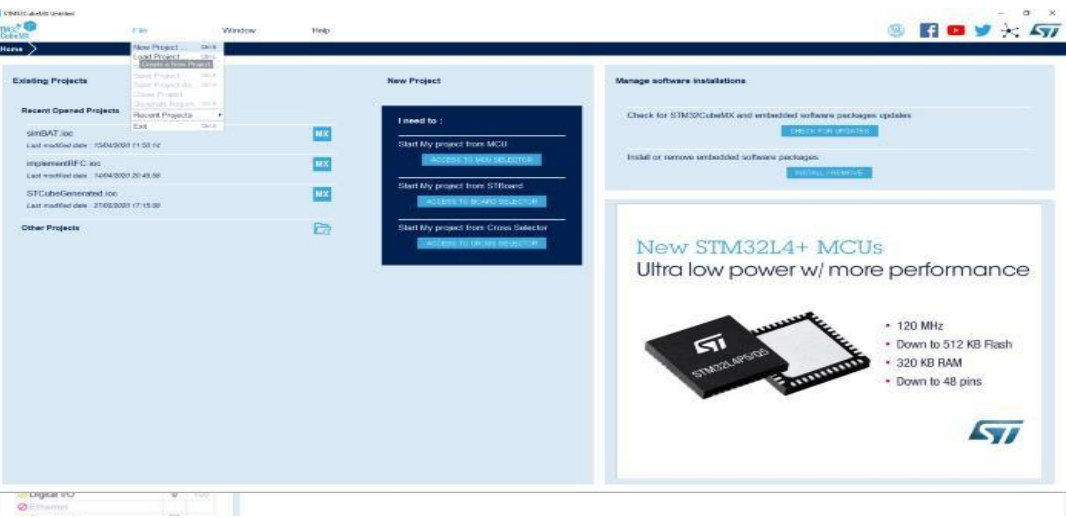


ARM is a 32-bit RISC (Reduced Instruction Set Computing) architecture. 32-bit implies that there are 32-lines in the address bus, i.e., it can address up to  $2^{32}$  locations. The RISC instruction implies that the hardware is complex, and a lot of process handling is done by the hardware, thus making it easier to code instructions for ARM processors.

### The STM32F4-Discovery:

Since STM32 is an ARM based microcontroller, it will have distinctions based on whether it is Core 7, Core 4, Core 0 etc. In this blog, I have made use of a development kit ie; STM32F4 Discovery board. A dev-kit, as can be guessed from the name, is used when developing an application. It is essentially a break-out board, which gives access to almost all the pins on the STM32 IC and has a built-in accelerometer and magnetometer. Thus, it is a great tool to develop and test your application/firmware. Once it can be ensured that the code is good and working, it can then be ported to an actual IC on a production grade PCB. The STM32F4 dev-kit has the STM32F407VGTx as the on-board IC. If we go through the datasheet, the IC part number gives the specific details and capabilities of the IC.

## Steps to setup a Project using Cube MX:



These pins which are selected will appear in green on the board selector. Once the peripheral configurations are done, go to the **Project Manager** tab.

## Coding and Debugging:

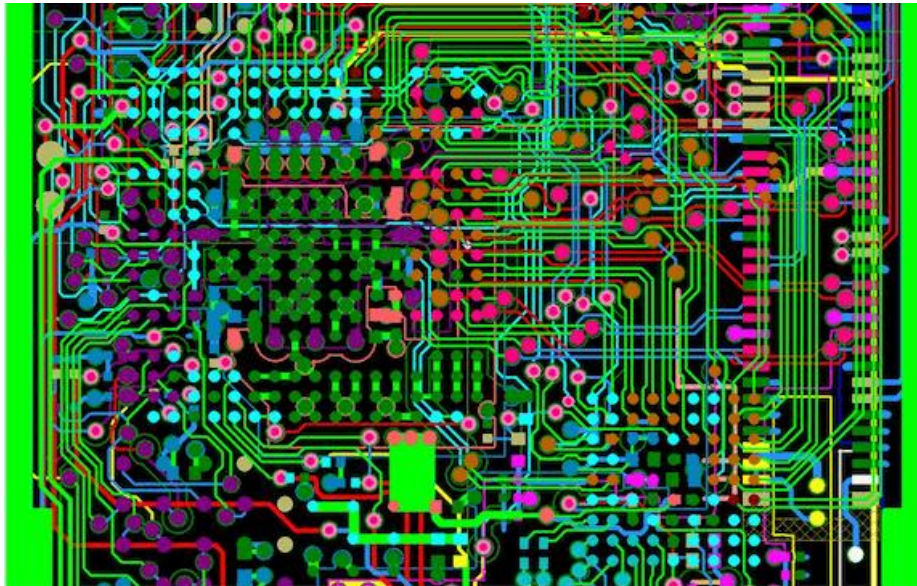
With the code generated, we can start using the peripherals and GPIO pins access directly in the main function. The code which runs on a loop is to be included in the while loop. Once you are done with the functional coding, we need to debug, i.e., test and run the code on the discovery board.

19

## **PCB Designing:**

Printed circuit board (PCB) design brings your electronic circuits to life in the physical form. Using layout software, the PCB design process combines component placement and routing to define electrical connectivity on a manufactured circuit board.

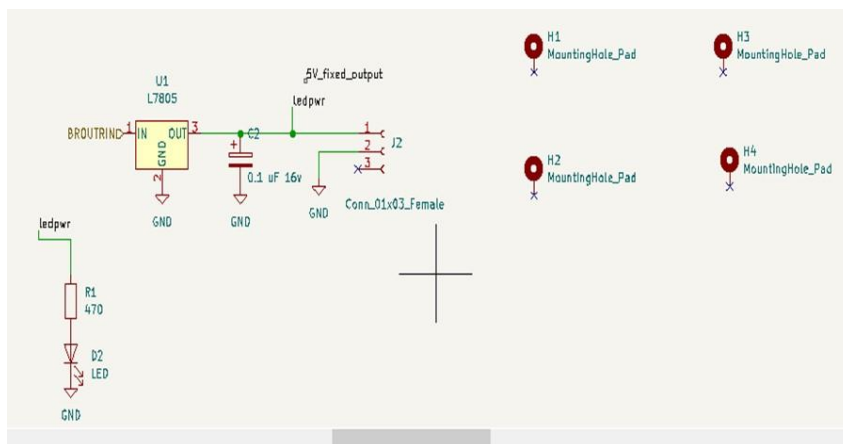
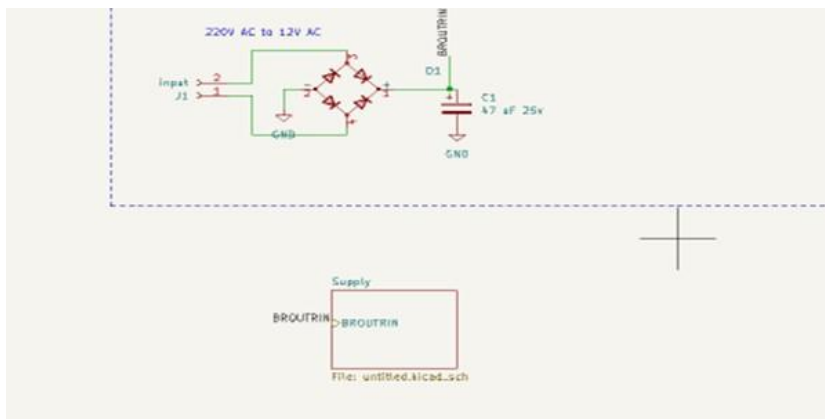
KiCad is a free software suite for electronic design automation (EDA). It facilitates the design and simulation of electronic hardware. It features an integrated environment for schematic capture, PCB layout, manufacturing file viewing, SPICE simulation, and engineering calculation.



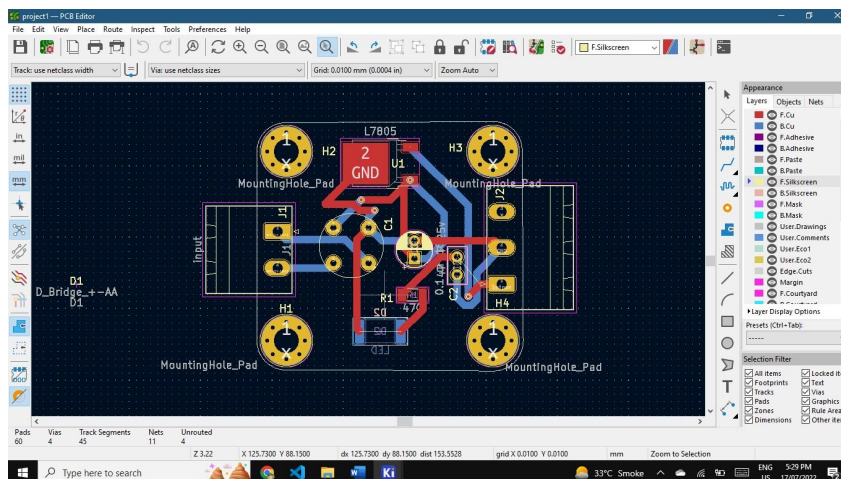
### **Schematic editing**

The KiCad schematic editor has features including hierarchical schematic sheets, custom symbol creation, ERC (electrical rules check) and integrated ngspice circuit simulation. Schematic symbols are very loosely coupled to circuit board footprints to encourage reuse of footprints and symbol.

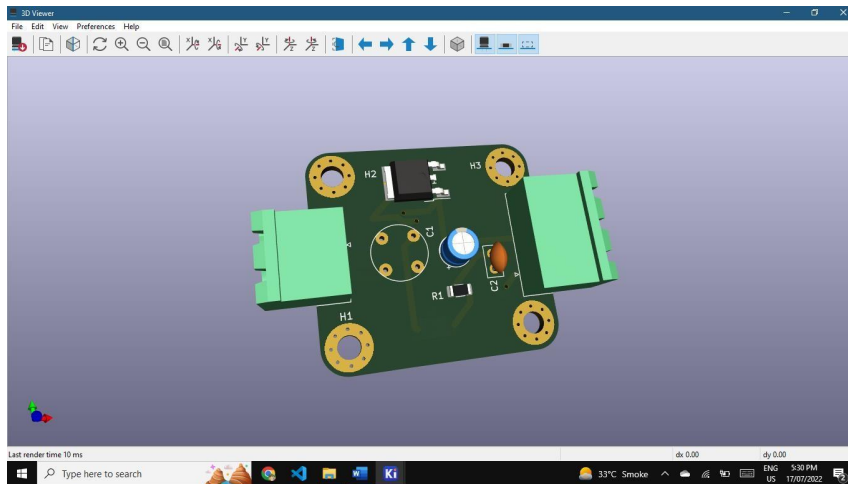
## Schematic Circuit of Power Supply Circuit:



## PCB Footprint:



### 3D view of circuit:



### Conclusion:

During my internship at EPTeck Technologies, I have learnt a lot about advanced embedded systems and IOT. The experience was great and I learnt thing from scratch. Our instructor “**Sir Waqas**” guided us briefly through every single concept of embedded systems in simple and easy to understand manner. I am confident enough that after this internship I am able to perform as an advance embedded developer. Special thanks to “**Sir Amir Zahoor**” for providing this opportunity of learning.

