Table of Contents

Microcontroller vs Microprocessor

Microcontroller

- Microcontroller is the heart of embedded system.
- Microcontroller has external processor along with internal memory and I/O ports.
- Since memory and I/O are present internally, circuit is small.
- Cost of entire system is low.
- Since components are internal, most operations are internal instruction hence speed is fast.

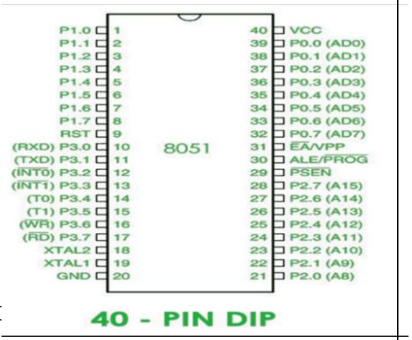
Microprocessor

- Microprocessor is heart of computer system.
- It is just a processor. Memory and I/O has to be connected externally.
- Since memory and I/O has to be connected externally, circuit becomes large.
- Cost of entire system is high.
- Since memory and I/O components are external each instruction perform external operation hence speed is slow.

8051 vs ARM CORTEX

8051

- 8 bit for standard core bus width is present in 8051 microcontroller.
- Speed is 12 clock cycles per machine cycle.
- UART, USART, I2C, SPI, communication protocols are used.
- Flash, ROM, SRAM memory is used in 8051 microcontroller.
- Uses CISC architecture.



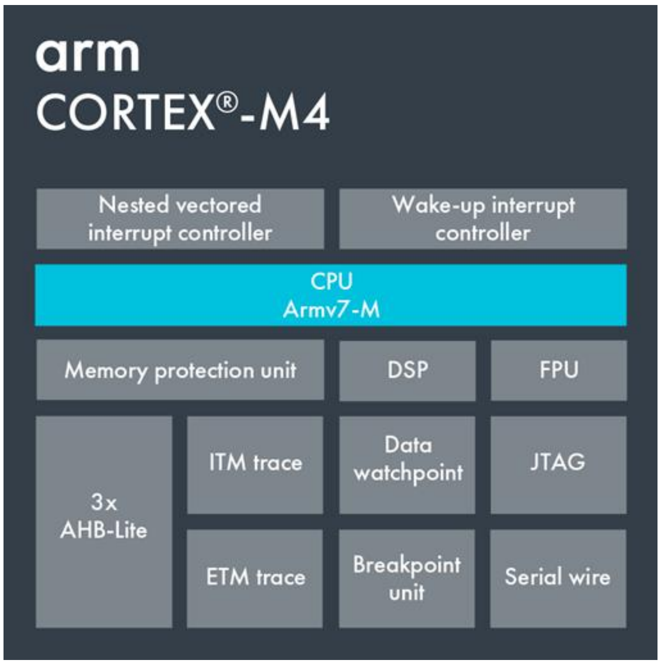
ARM CORTEX

- 32 bit is present in ARM microcontroller and 64 bit is also availal
- Speed is 1 clock cycle per machine cycle.
- UART, USART, Ethernet, I2S, DSP, SPI, CAN, LIN, 12C, communication protocols are used.
- Flash, EPROM, SDROM, memory is used in ARM microcontroller.
- USES RISC architecture.

Why ARM?

- Due to its low power mode, efficiency, low latency and accumulates operation with high precision.

Architecture of ARM CORTEX



Pipelining- Decoding is easier (decode in one stage). Mechanism for overlapped execution, used in instruction, arithmetic, memory access.

FPU (Floating point unit) -

32 bit single precision, it has IEEE rounding modes, Performs operations as PUSH, POP.

To accumulate the operation, it has high precision. It can perform operations like +, -, *, /, square root.

JTAG -

Serial wire of JTAG or JTAG debug interface is used for debugging, mainly using printf()

Interrupts- (Nested vectored Interrupt controller) -

Low latency. Also it has (1 to 240) External interrupts.

3 to 8 Priority bits.

Selection of preempting levels and non preempting levels

Serial wire, breakpoint wire -

It comes in debug features.

Serial wire - (SWDP) and (SWJDP) implementing breakpoint and patches.

MPU (Memory protection unit) -

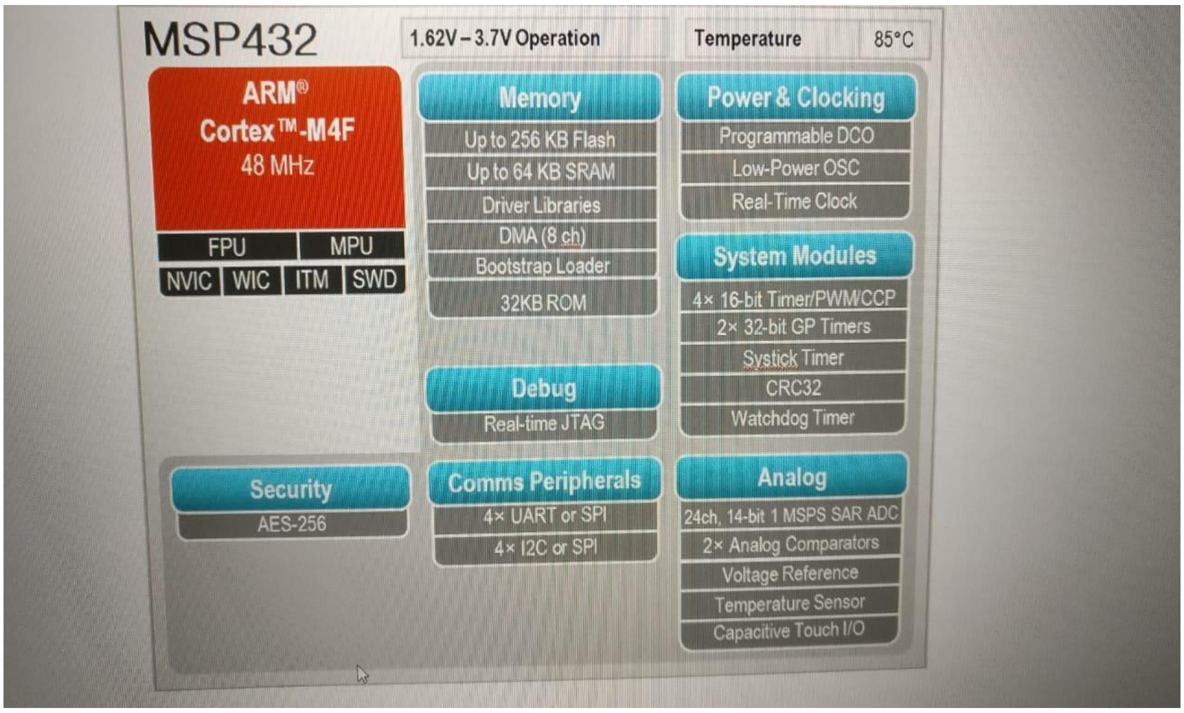
8 Memory regions.

Sub regions disable (SRD) which helps in efficient usage of memory.

Architecture of MSP432 built on top of ARM M4 architecture.

- It is 32 bit processor family, mixed signal processor.

    - ARM 32 bit cortex M4F CPU with floating point unit and memory protection unit.
    - Frequency upto 48MHz
    - Memories- upto 256KB of flash main memory; upto 64KB of SRAM; 32 KB of ROM with MSP432 peripheral driver libraries.



5.2 Memory: ROM, flash memory and SRAM

MSP32 ROM: Read Only Memory.

- Data stored in these chips is non-volatile (type of memory that can retain stored information even after power is removed).
- Data stored in these chip is either unchangeable or requires special operation to change (unlike RAM which can be changed as easily it can be read.
- It normally allows current to flow in only one direction & has a certain threshold known as FORWARD BREAKOVER(determines how much current is required before the diode will pass)

  The types are-

  a. *PROM* (Programmable Read Only Memory) - Creating a ROM is time consuming and expensive, for that developers created PROM. It can be easily coded with special tool called programmer. It can be programmed once.

  b. *EPROM* (Erasable Programmable Read Only Memory) - It can be rewritten many times. It requires special tool that emits a certain frequency of UV light.

  c. *EEPROM* (Electrically Erasable Programmable Read Only Memory) – In this, the chip does not have to be removed or rewritten. Instead of using UV light, we can return the electrons to EEPROM & they are changed 1 byte at a time which makes them versatile but slow.

FLASH MEMORY: A type of EEPROM, but also can read data quickly without power loss. It is non-volatile. It can electrically be erased and reprogrammed. They work much faster than EEPROM's.

Flash memory is a non-volatile memory chip used for storage and for transfering data between a personal computer (PC) and digital devices. It has the ability to be electronically reprogrammed and erased. It is often found in USB flash drives, MP3 players, digital cameras and solid-state drives.

-RAM: Random Access Memory (volatile memory), it holds different type of program data such as temporary variables, global variables. It is lesser in size than flash memory and low latency than flash memory.

SRAM **(**Static Random Access Memory**) :** It is very fast and is the fastest storage device to read & write currently, & is very expensive (uses latching circuitry flip-flop to store each bit). It has high power consumption.

Power clocking: RTC, low power OSC, programmable DCO

-RTC (Real time clock):

- Is inbuilt component available. It provides seconds, minutes, hours, day of weeks, day of month & year.
- Interrupt capability (returns the number of interrupt).
- Operates in low power mode.
- Binary format.
- As the oscillator gets supply it starts processing.
- External oscillator is must.

- **Low power OSC:** It is used in generating and receiving RF signals where a variable frequency is required.

Programmable DCO (DIGITAL CONTROLLED OSCILLATOR)-

This oscillator can be used for everything from setting baud rates for a UART or for setting the clock.

5.4 System modules: Watchdog timer, Systick timer, PWM

-Watchdog timer:

- Used to regain control when system failed because of software error, reset when time out value is reached.
- It is non-maskable interrupt.
- It is a timer which is used to detect & recover from deadlock conditions.

-Systick timer:

- It is 24 bit countdown timer with an auto reload (when timer gets started, then the value gets automatically loaded)
- The default clock source for systick timer is cortex-M CPU clock.
- It helps in generating accurate interrupts to different task(of RTOS).

PWM (Pulse Width Modulation):

- It can generate signals of varying frequency and duty cycle on one or more output pins.
- Commonly used to control the speed of electric motors, brightness of light.

Analog: Capacitive touch I/O, temp sensor, ADC, analog comparators.

- **Capacitive touch sensors:** the electrode represents one of the plates of the capacitor. The output capacitance will increase if a conductive object touches or approaches the sensor electrode. The measurement circuit will detect the change in the capacitance and converts it into a trigger signal.

- **Temperature Sensor:** is a device used to measure temperature. This can be air temperature, liquid temperature or temperature of solid matter. Different types are,

a. *Thermistors:* can be very small in size. They measure temperature by measuring the change in resistance of the electric current. Thermistors are available as either NTC or PTC and are often low cost.

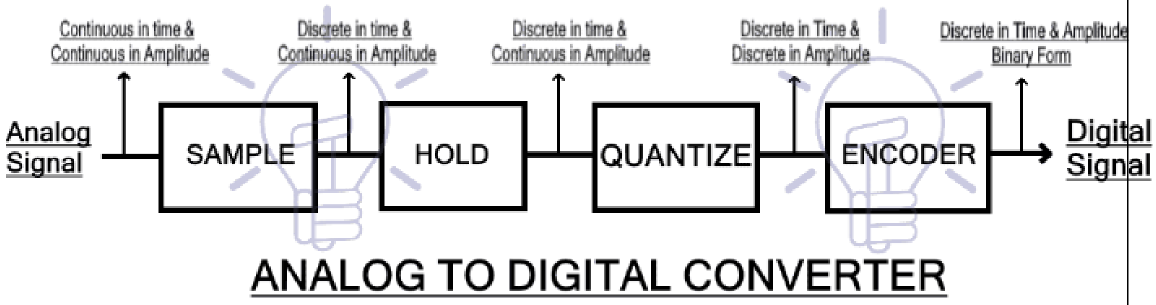
b. *RTDs or Resistance Temperature Detectors*: work in a similar way to Thermistors and measure ohmic resistance to measure temperature. They are connected to a circuit in a similar way to a thermistor but they have a much wider temperature range and can measure extreme temperatures.

c. *Thermocouples:* use two conductors, made up of different metals that are joined at the end to form a junction. When this junction is subjected to heat, a voltage is produced that is directly proportional to the temperature input.

d. *Temperature Probes:* are a very common and diverse type of temperature sensor. They consist of either a thermistor, a thermocouple or RTD sensing element and can be finished with a terminal head.

ADC:

- It converts an analog signal (sound, light) into digital.
- It converts an analog voltage/current to a digital number.
- ADCs follow a sequence when converting analog signals to digital.
- They first sample the signal, then quantify it to determine the resolution of the signal, and finally set binary values and send it to the system to read the digital signal.
- Two important aspects of the ADC are its sampling rate and resolution.


Continuous in time & Continuous in Amplitude
Discrete in time & Continuous in Amplitude
Discrete in time & Continuous in Amplitude
Discrete in Time & Discrete in Amplitude
Discrete in Time & Amplitude Binary Form
Analog Signal
SAMPLE
HOLD
QUANTIZE
ENCODER
Digital Signal
ANALOG TO DIGITAL CONVERTER

Sample-

The sample block function is to sample the input analog signal at a specific time interval. The samples are taken in continuous amplitude & possess real value but they are discrete with respect to time.

Hold-

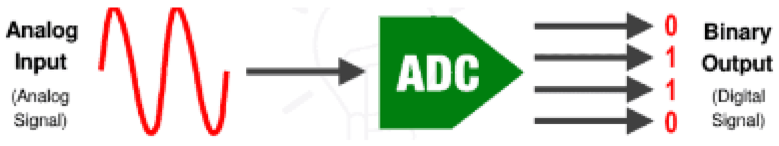
The second block used in ADC is the ‘Hold’ block. It has no function. It only holds the sample amplitude until the next sample is taken. The hold value remains unchanged till the next sample.

Quantize-

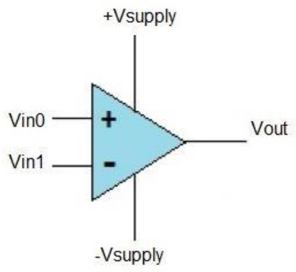
This block is used for quantization. It converts the analog or continuous amplitude into discrete amplitude. The on hold continuous amplitude value in hold block goes through 'quantize' block & becomes discrete in amplitude. The signal is now in digital form as it has discrete time & discrete amplitude.

Encoder-

The encoder block converts the digital signal into binary form i.e. into bits. We know that the digital devices operate on binary signals so it is necessary to convert the digital signal into the binary form using the Encoder.



- ANALOG COMPARATOR: The Analog Comparator is a circuit which compares two analog voltages available at the inputs and generates output based on which one of the inputs is greater/lesser to the othe



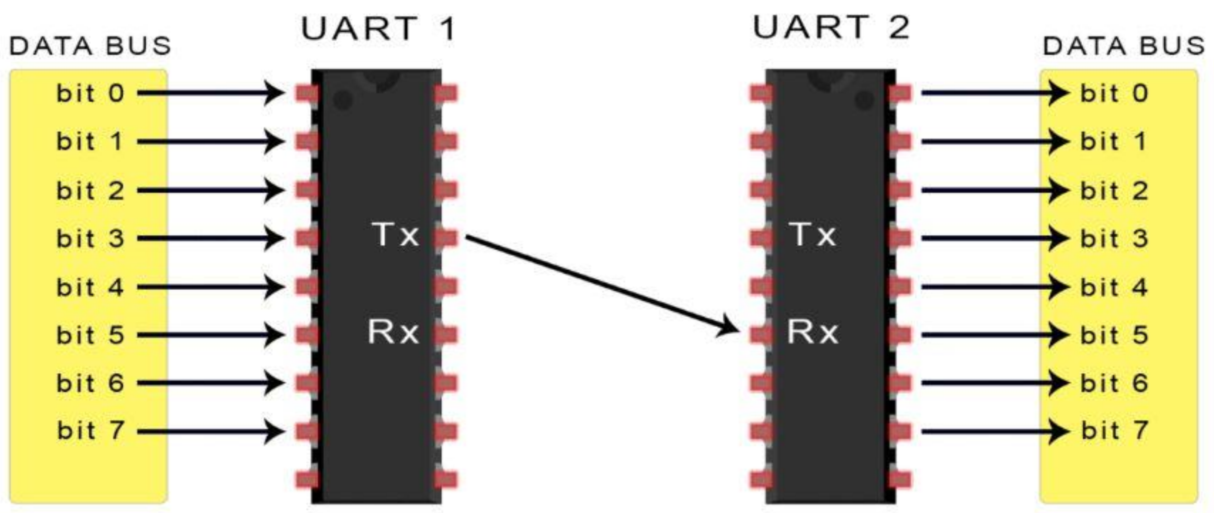
When Vin0>Vin1, Vout is equal to +Vsupply

When Vin0<Vin1, Vout is equal to –Vsupply

Peripheral interface: I2C, SPI, UART

- UART- Universal Asynchronous Receiver Transmitter. It has high resolution baud rate

In UART communication, two UARTs communicate directly with each other. The transmitting UART converts parallel data from a controlling device like a CPU into serial form, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving device. Only two wires are needed to transmit data between two UARTs. Data flows from the Tx pin of the transmitting UART to the Rx pin of the receiving UART.



UART transmitted data is organized into packets. Each packet contains 1 start bit, 5 to 9 data bits (depending on the UART), an optional parity bit, and 1 or 2 stop bits

START BIT

The UART data transmission line is normally held at a high voltage level when it’s not transmitting data. To start the transfer of data, the transmitting UART pulls the transmission line from high to low for one clock cycle. When the receiving UART detects the high to low voltage transition, it begins reading the bits in the data frame at the frequency of the baud rate.

DATA FRAME

The data frame contains the actual data being transferred. It can be 5 bits up to 8 bits long if a parity bit is used. If no parity bit is used, the data frame can be 9 bits long. In most cases, the data is sent with the least significant bit first.

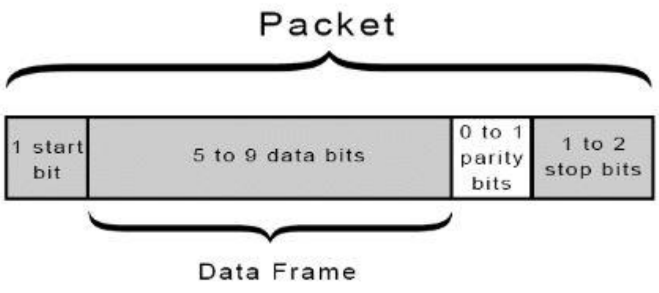
PARITY

Parity describes the evenness or oddness of a number. The parity bit is a way for the receiving UART to tell if any data has changed during transmission. Bits can be changed by electromagnetic radiation, mismatched baud rates, or long distance data transfers. After the receiving UART reads the data frame, it counts the number of bits with a value of 1 and checks if the total is an even or odd number. If the parity bit is a

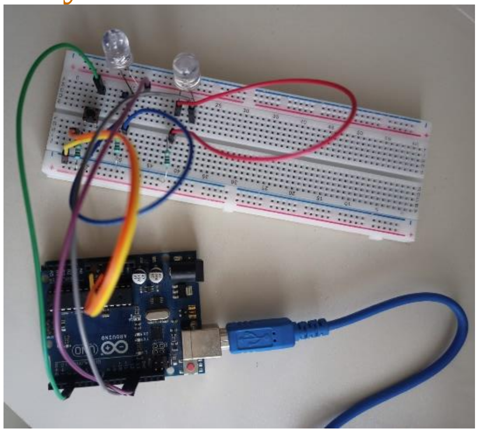
0 (even parity), the 1 bits in the data frame should total to an even number. If the parity bit is a 1 (odd parity), the 1 bits in the data frame should total to an odd number. When the parity bit matches the data, the UART knows that the transmission was free of errors. But if the parity bit is a 0, and the total is odd; or the parity bit is a 1, and the total is even, the UART knows that bits in the data frame have changed.

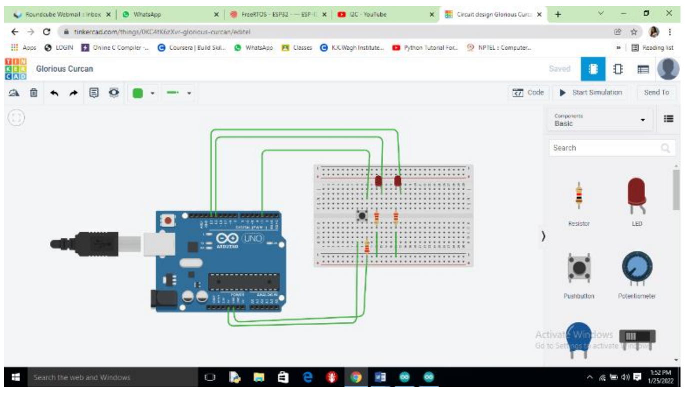
STOP BITS

To signal the end of the data packet, the sending UART drives the data transmission line from a low voltage to a high voltage for at least two bit durations.



UART By Button Press:





https://drive.google.com/drive/folders/1esRbDQsArwLatvz07JWen8HB1XehvjrN

{This google drive link is provided for practically implemented videos }

```
int LED1 = 12;

int LED2 = 13;

int button = 3;


boolean LED1State = false;

boolean LED2State = false;
```

```
long buttonTimer = 0;
long longPressTime = 350;

boolean buttonActive = false;
boolean longPressActive = false;

void setup() {

  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(button, INPUT);

}

void loop() {

  if (digitalRead(button) == HIGH) {

    if (buttonActive == false) {

      buttonActive = true;
      buttonTimer = millis();

    }

    if ((millis() - buttonTimer > longPressTime) && (longPressActive == false)) {

      longPressActive = true;
```

```
        LED1State = !LED1State;
        digitalWrite(LED1, LED1State);

      }

    } else {

      if (buttonActive == true) {

        if (longPressActive == true) {

          longPressActive = false;

        } else {

          LED2State = !LED2State;
          digitalWrite(LED2, LED2State);

        }

        buttonActive = false;
  }
  }
  }
```

- Ft232RL TTL Serial Adaptor: TTL UART MODULE used for SERIAL COMMUNICATION. FT232RL is the IC used on UART MODULE

Pinout-

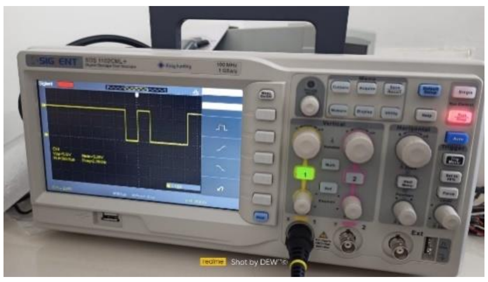
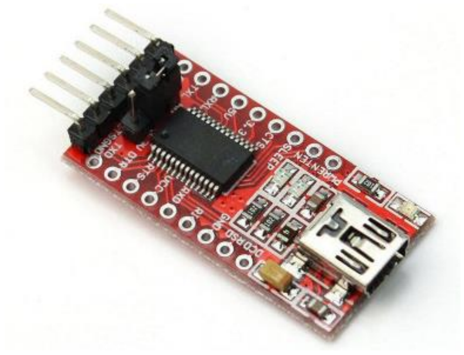
1) DTR (Data Terminal Ready)- output pin used for flow control

2) RX(Receiver)

3) TX(Transmitter)

4) VCC- positive voltage output, controlled by the jumper

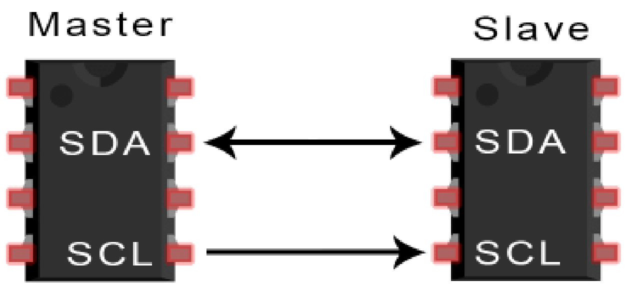
5) CTS (Clear to send)- input pin used for flow control   6) GND

I2C-

I2C stands for Inter-Integrated Circuit. It is a bus interface connection protocol incorporated into devices for serial communication. It is a synchronous, multi-master, multi-slave, packet switched.

SDA (Serial Data) – The line for the master and slave to send and receive data.
SCL (Serial Clock) – The line that carries the clock signal.



With I2C, data is transferred in messages. Messages are broken up into frames of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted. The message also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame-

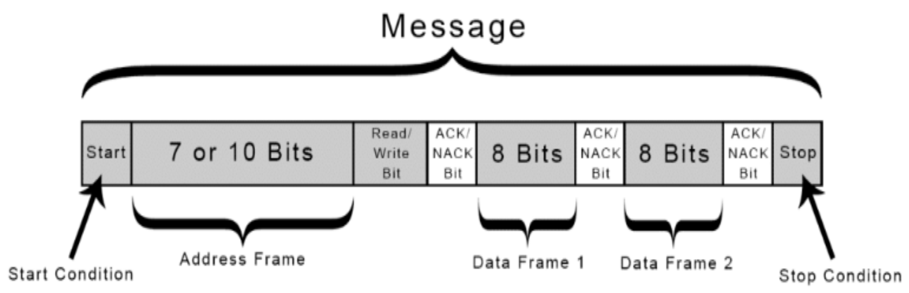
Start Condition: The SDA line switches from a high voltage level to a low voltage level before the SCL line switches from high to low.

Stop Condition: The SDA line switches from a low voltage level to a high voltage level after the SCL line switches from low to high.

Address Frame: A 7- or 10-bit sequence unique to each slave that identifies the slave when the master wants to talk to it.

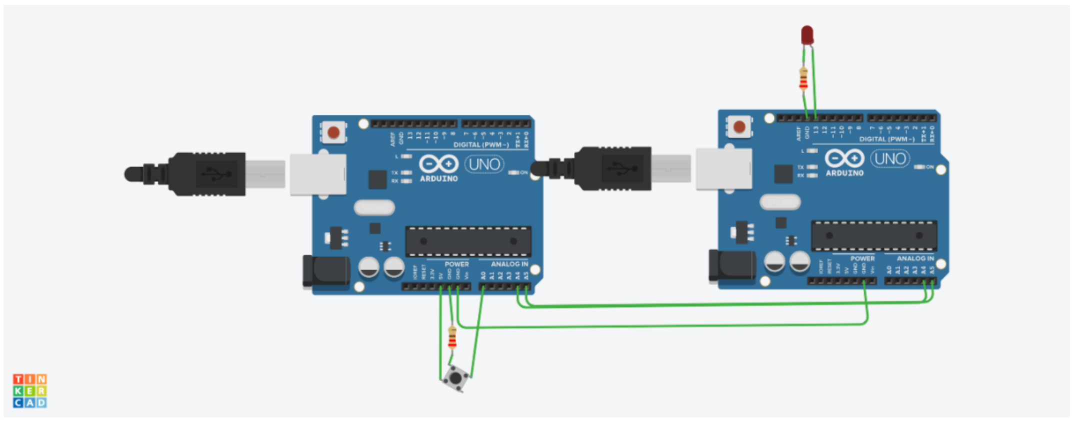
Read/Write Bit: A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level).

ACK/NACK Bit: Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.



I2C Libraries:

1. Wire.begin() - Initiate the Wire library and join the I2C bus as a master or slave. This should normally be called only once.
   Address: the 7-bit slave address (optional); if not specified join the bus as a master.
2. Wire.requestFrom() - Used by master to request bytes from slave device. The bytes may be retrieved by available() & read().
3. Wire.onReceive() – Registers a function to be called when a slave device receives a transmission from master.
4. Wire.beginTransmission() – Begins transmission to I2C slave device with given address.
5. Wire.endTransmission() – Ends transmission to slave device that was begin.
6. Wire.available() – Returns the number of bytes available for retreival with read(). Should be called on master device after a call to requestFrom() or on slave inside onReceive().
7. Wire.read() – Reads a byte that was transmitted from slave device to a master after a call to requestFrom()
8. Wire.write() – Writes data from slave device in response to request from master.
9. Wire.setClock() – Function modifies the clock frequency for I2C communication.
10. Wire.onRequest(handler) – Register a function to be called when master requests data from slave device.



TRANSMITTER CODE	RECEIVER CODE

```
#include<Wire.h>
 int pushButton=A0;
 int x=0;
 void setup()
 {
   Wire.begin();
   pinMode(pushButton,
 INPUT);
 }

 void loop()
 {
   Wire.beginTransmission(1);
   x=digitalRead(pushButton);
   Wire.write(x);
   Wire.endTransmission();
   delay(500);
 }
```
	```
	#include<Wire.h>
	int Led=13;
	int x=0;
	void setup()
	{
	  Wire.begin(1);
	  Wire.onReceive(receiveEvent);
	  pinMode(Led, OUTPUT);
	}

	void loop()
	{

	  delay(100);
	}
	void receiveEvent(int howMany)
	{
	  x=Wire.read();
	  if(x==1){
	    digitalWrite(Led,HIGH);
	  }
	  else {
	    digitalWrite(Led,LOW);
	  }
	}
	```

https://drive.google.com/drive/folders/1esRbDQsArwLatvz07JWen8HB1XehvjrN

{This drive link is provided for practically implemented videos}

- SPI- The Serial Peripheral Interface (SPI) is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded system.

SPI devices communicate in full duplex mode using a master-slave architecture usually with a single master. The master (controller) device originates the frame for reading and writing. Multiple slave-devices may be supported through selection with individual chip select (CS), sometimes called slave select (SS) lines.

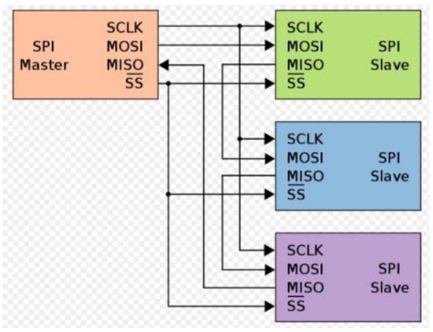
The SPI bus specifies four logic signals:

SCLK: Serial Clock (output from master)

MOSI: Master Out Slave In (data output from master)

MISO: Master In Slave Out (data output from slave)

CS /SS: Chip/Slave Select (often active low, output from master to indicate that data is being sent)



Advantages:

- Full duplex communication.
- Provide good signal integrity and high speed.
- Higher throughput than I²C.Not limited to any maximum clock speed, enabling potentially high speed.
- Extremely simple hardware interfacing
- Typically lower power requirements than I²C due to less circuitry (including pull up resistors)
- Slaves do not need a unique address – unlike I²C
- Uses only four pins on IC packages, and wires in board layouts or connectors, much fewer than parallel interfaces.
- At most one unique bus signal per device (chip select), all others are shared

Disadvantages-

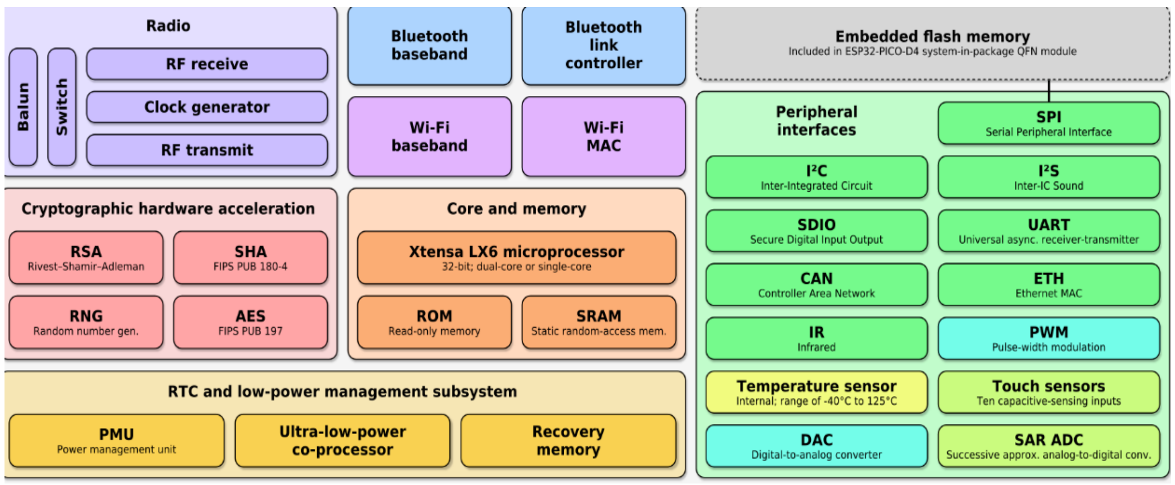
- Requires more pins on IC packages than I²C.
- Typically supports only one master device .
- No error-checking protocol is defined.

Applications-

- The full-duplex capability makes SPI very simple and efficient for single master/single slave applications.
- Sensors: temperature, pressure, ADC, touchscreens, video game controllers
- Control devices: digital potentiometers, DAC
- Camera lenses: Canon EF lens mount
- Communications: Ethernet, USB, USART, CAN, IEEE 802.15.4, IEEE 802.11, handheld video games
- Memory: flash and EEPROM
- Real-time clocks
- LCD, sometimes even for managing image data

ESP32 block diagram

ESP32


Radio
Balun
Switch
RF receive
Clock generator
RF transmit
Bluetooth baseband
Bluetooth link controller
Wi-Fi baseband
Wi-Fi MAC
Embedded flash memory
Included in ESP32-PICO-D4 system-in-package QFN module
Peripheral interfaces
SPI
Serial Peripheral Interface
I²C
Inter-Integrated Circuit
I²S
Inter-IC Sound
SDIO
Secure Digital Input Output
UART
Universal async. receiver-transmitter
CAN
Controller Area Network
ETH
Ethernet MAC
IR
Infrared
PWM
Pulse-width modulation
Temperature sensor
Internal; range of -40°C to 125°C
Touch sensors
Ten capacitive-sensing inputs
DAC
Digital-to-analog converter
SAR ADC
Successive approx. analog-to-digital conv.
Cryptographic hardware acceleration
RSA
Rivest-Shamir-Adleman
SHA
FIPS PUB 180-4
RNG
Random number gen.
AES
FIPS PUB 197
Core and memory
Xtensa LX6 microprocessor
32-bit; dual-core or single-core
ROM
Read-only memory
SRAM
Static random-access mem.
RTC and low-power management subsystem
PMU
Power management unit
Ultra-low-power co-processor
Recovery memory

ESP32 is a series of low-cost, low-power system on a chip microcontroller with integrated Wi-Fi and dual-mode Bluetooth. Features include-

Processors:

CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz

Ultra-low power (ULP) co-processor

Memory: 320 KiB RAM, 448 KiB ROM

Wireless Connectivity:

Wi-Fi: 802.11 b/g/n

Bluetooth: v4.2 BR/EDR and BLE (shares the radio with Wi-Fi)

Peripheral interfaces:

34 × programmable GPIOs

12-bit SAR ADC up to 18 channels

2 × 8-bit DACs

10 × touch sensors (capacitive sensing GPIOs)

4 × SPI

2 × I²S interfaces

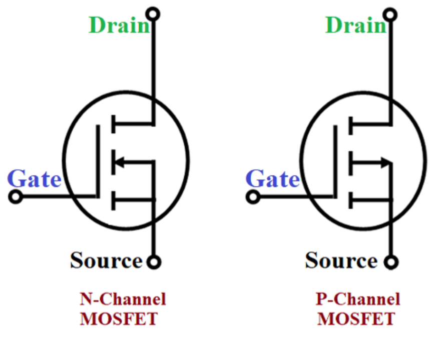
2 × I²C interfaces

3 × UART

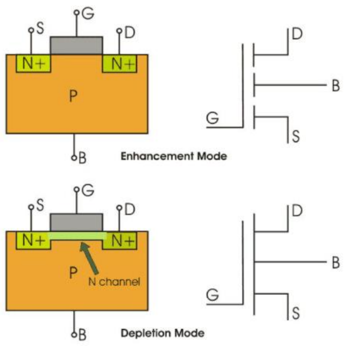
-Read about relay, MOSFET, transistor working, usage.

- MOSFET: Metal Oxide Field Effect Transistor. MOSFETs are also known as IGFET (Insulated Gate Field Effect Transistor) as the gate pin is electrically insulated from the semiconductor.

MOSFET is a voltage-controlled device, as opposed to BJT (bipolar junction transistor) that is a current-controlled device.



- The MOSFET is a voltage-controlled device where current flows between drain and source terminal when voltage is applied at the gate terminal.
- In the case of N-Channel MOSFET, when a positive voltage is applied at the gate pin, it produces the conduction process between Drain & Source terminals.
- In this condition, MOSFET is considered to act in an Enhancement Mode. The positive voltage at the Gate terminal will attract the electrons below the oxide layer and repel the holes. As a result, the electrons will start accumulating below the silicon oxide layer.
- The more positive voltage at the gate terminal will attract more electrons and thus conduction process increases in N-Channel MOSFET.
- If we apply negative voltage at the gate terminal, it will force the N-type MOSFET to repel electrons and attract holes. As a result, there will be no connection between Drain & Source terminals. In this condition, MOSFET stands in the Depletion Mode.

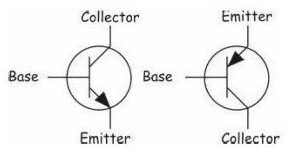


Relay:

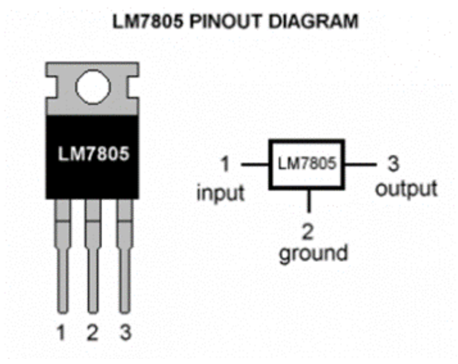
- A relay is an electrically operated switch. It consists of a set of input terminals for a single or multiple control signals, and a set of operating contact terminals.
- Relay works on the principle of electromagnetic induction. When the electromagnet is applied with some current it induces a magnetic field around it.
- Relays are generally used to switch smaller currents in a control circuit and do not usually control power consuming devices.
- When the circuit of the relay senses the fault current, it energises the electromagnetic field which produces the temporary magnetic field.

Transistor:

- A circuit that has low resistance participates in the transfer of the signals that are weak to the circuit with the high resistance.
- A transistor is an electronic component that is used in circuits to either amplify or switch electrical signals or power. A transistor consists of two PN diodes, it has three terminals namely emitter, base and collector.

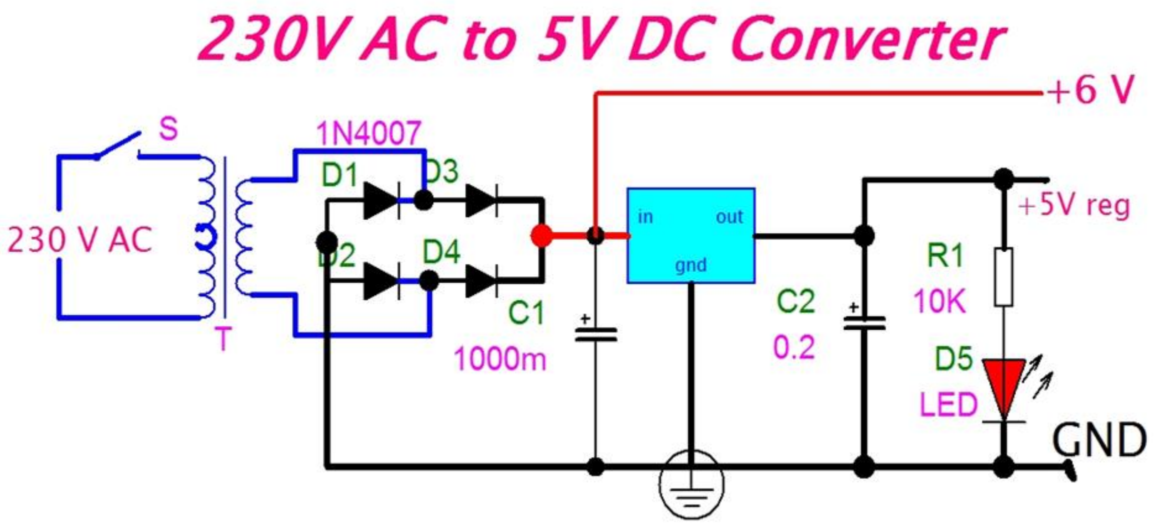


1) **12-24v to 5v DC conversion:** IC used is LM7805 (Input- 7 to 25 V, Output- 5V)



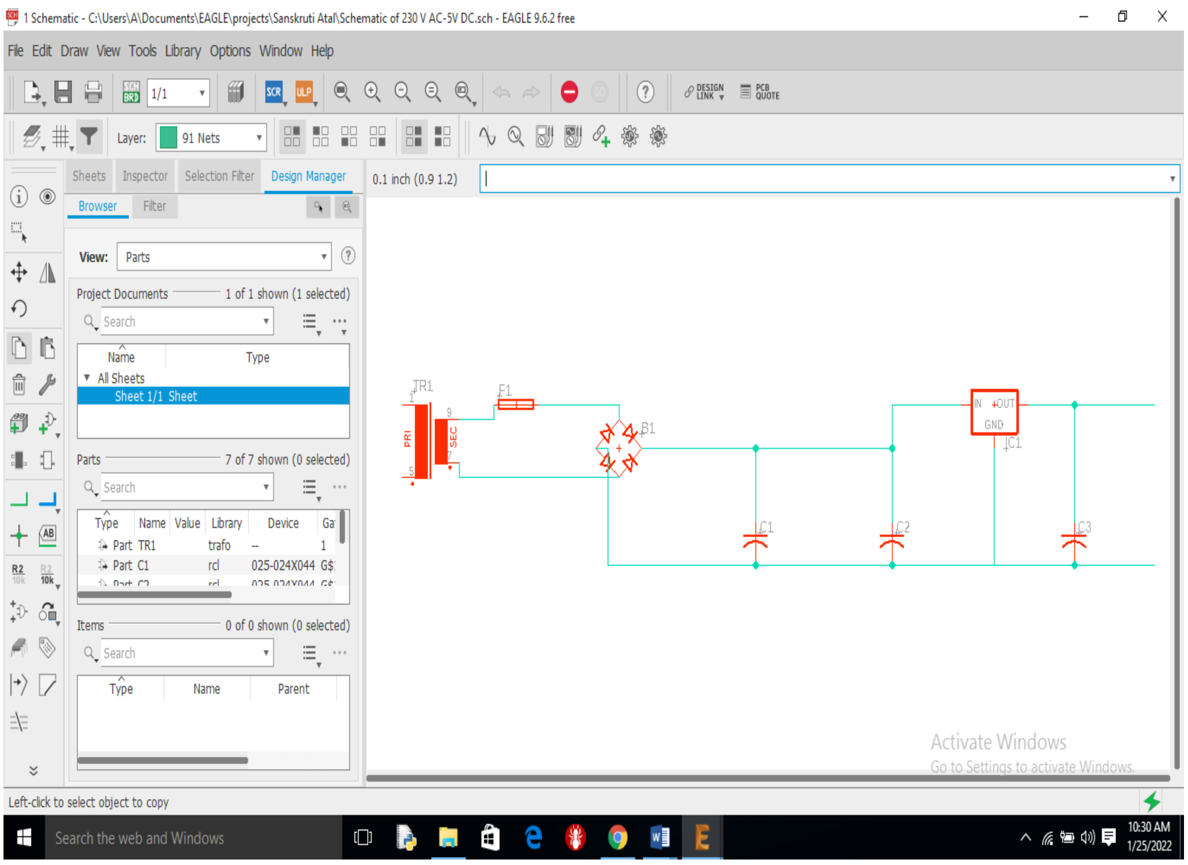
2) 12-24v to 8v DC: IC used is LM7808 (Input- 10.5 to 25 V, Output- 8V)

3) 230v AC to 5v DC conversion:(Step down transformer, DB107 bridge rectifier, LM7805)



Circuit designing by using eagle software:

230v AC to 5v DC conversion using Eagle software and testing.


1 Schematic - C:\Users\A\Documents\EAGLE\projects\Sanskruti Atal\Schematic of 230 V AC-5V DC.sch - EAGLE 9.6.2 free

Layer: 91 Nets

0.1 inch (0.9 1.2)

View: Parts

Project Documents 1 of 1 shown (1 selected)

Name	Type
All Sheets	
Sheet 1/1	Sheet

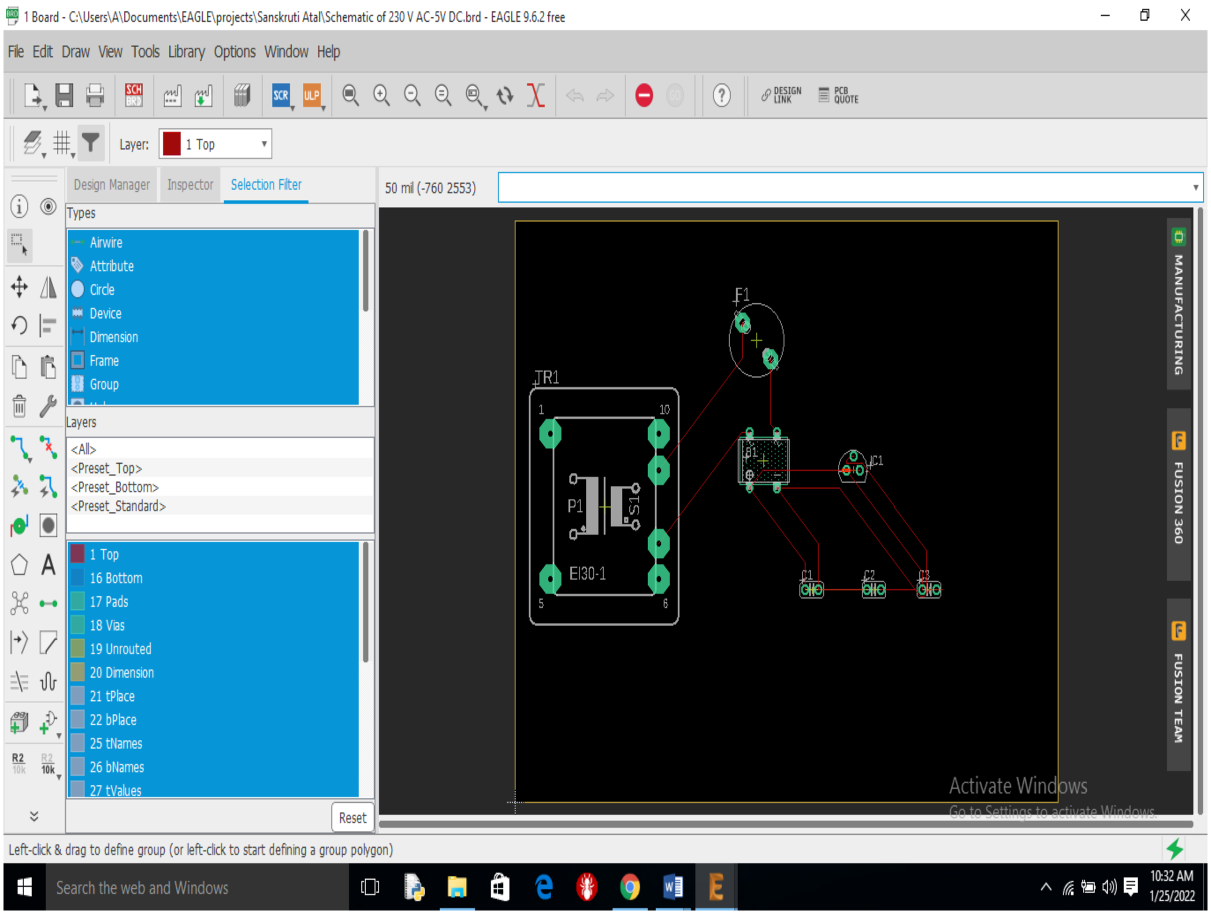
Parts 7 of 7 shown (0 selected)

Type	Name	Value	Library	Device	Ga
Part	TR1		trafo	--	1
Part	C1		rcl	025-024X044	G$

Items 0 of 0 shown (0 selected)

Type	Name	Parent

Left-click to select object to copy

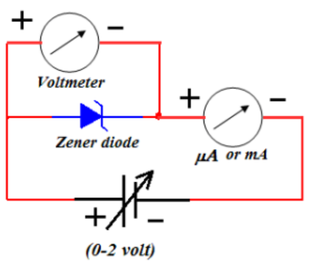

1 Board - C:\Users\A\Documents\EAGLE\projects\Sanskruti Atal\Schematic of 230 V AC-5V DC.brd - EAGLE 9.6.2 free
File Edit Draw View Tools Library Options Window Help
Layer: 1 Top
Design Manager
Inspector
Selection Filter
Types
Airwire
Attribute
Circle
Device
Dimension
Frame
Group
Layers
<All>
<Preset_Top>
<Preset_Bottom>
<Preset_Standard>
1 Top
16 Bottom
17 Pads
18 Vias
19 Unrouted
20 Dimension
21 tPlace
22 bPlace
25 tNames
26 bNames
27 tValues
Reset
50 mil (-760 2553)
F1
TR1
EI30-1
P1
S1
B1
IC1
C1
C2
C3
MANUFACTURING
FUSION 360
FUSION TEAM
Activate Windows
Go to Settings to activate Windows.
Left-click & drag to define group (or left-click to start defining a group polygon)
Search the web and Windows
10:32 AM
1/25/2022

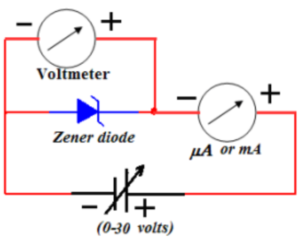
Find IC which can convert both DC and AC input to DC conversion.

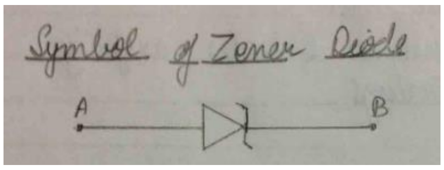
- We can use a bridge rectifier IC that contains 4 diodes, bridge rectifier IC has 4 pins: 2 for AC i/p, 2 for DC o/p.

Study VI characteristics of diode and Zener diode.

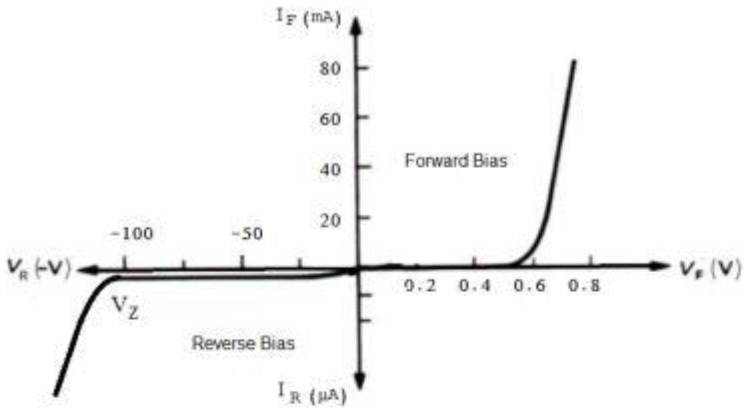
- Zener diode is a heavily doped PN junction diode. Due to heavily doped, its depletion layer is very thin.
- The forward bias characteristic of Zener diode is same as the normal PN junction diode but in reverse bias it has different characteristic.
- Initially, a negligible constant current flow through the zener diode in its reverse bias but at certain voltage, the current becomes abruptly large. This voltage is called as zener voltage.
- This sudden and sharp increase in zener current is called as zener breakdown.





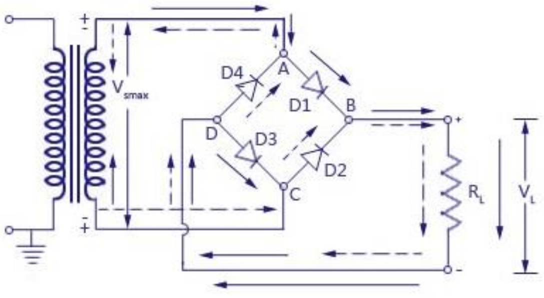


- **Avalanche Break down:** When p-region and n-region of a diode are lightly doped, depletion region at the junction will broaden. If a very large electric field applied to the junction, the kinetic energy of the charge carriers increases which collides with the adjacent atoms producing charge carriers by breaking the bonds. This process results in large current causing avalanche breakdown
- On increasing Vd, width of depletion layer decreases, thus barrier potential Vb decreases.
- When Vd=Vb, there is rapid increase in current.

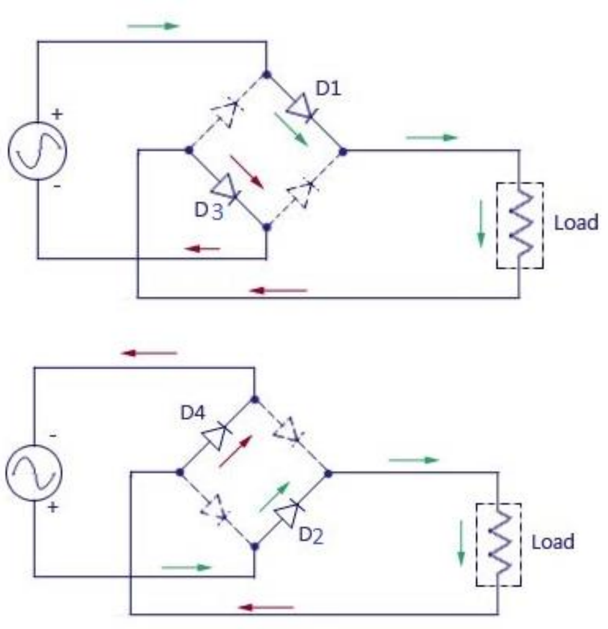


Full wave and bridge rectifier design.

- A Full wave rectifier is a circuit arrangement which makes use of both half cycles of input alternating current (AC) and converts them to direct current (DC).
- In the circuit diagram, 4 diodes are arranged in the form of a bridge. The transformer secondary is connected to two diametrically opposite points of the bridge at points A & C. The load resistance RL is connected to bridge through points B and D.



- During the first half cycle of the input voltage, the upper end of the transformer secondary winding is positive with respect to the lower end. Thus during the first half cycle diodes D1 and D3 are forward biased and current flows through arm AB, enters the load resistance RL, and returns back flowing through arm DC. During this half of each input cycle, the diodes D2 and D4 are reverse biased and current is not allowed to flow in arms AD and BC.
- During the second half cycle of the input voltage, the lower end of the transformer secondary winding is positive with respect to the upper end. Thus diodes D2 and D4 become forward biased and current flows through arm CB, enters the load resistance RL, and returns back to the source flowing through arm DA. The flow of current has been shown by dotted arrows in fig(2)
- Thus the direction of flow of current through the load resistance RL remains the same during both half cycles of the input supply voltage.

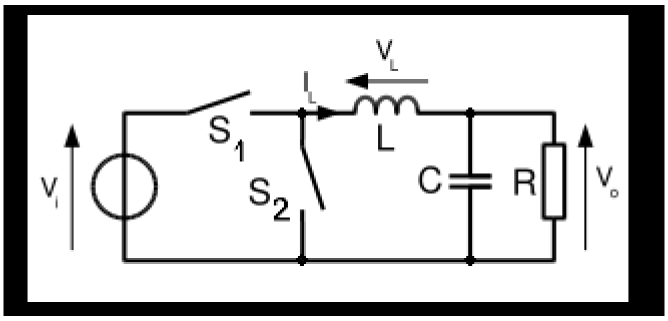


- **FreeRTOS:** It is free real time operating system.

```
#include<Arduino_FreeRTOS.h>
void Task_Print1(void *p);
void Task_Print2(void *p);

TaskHandle_t Task_Handle1;
TaskHandle_t Task_Handle2;

void setup() {
  Serial.begin(9600);
  xTaskCreate(Task_Print1,"Task1",100,NULL,2,&Task_Handle1);
  xTaskCreate(Task_Print2,"Task2",100,NULL,2,&Task_Handle2);

}
void loop() {
}
void Task_Print1(void *p)
{
  (void) p;
  while(1)
  {
    Serial.println("Task1");
    vTaskDelay(1000/portTICK_PERIOD_MS);
  }
}
void Task_Print2(void *p)
{
  (void) p;
  while(1)
  {
    Serial.println("Task2");
    vTaskDelay(1000/portTICK_PERIOD_MS);
  }
}
```

- **INTERRUPTS:** An interrupt is a signal to the processor emitted by hardware or software indicating an event that **needs immediate attention**. Whenever an interrupt occurs, the controller completes the execution of the current instruction and starts the execution of an Interrupt Service Routine (ISR) or Interrupt Handler. ISR tells the processor or controller what to do when the interrupt occurs.
  Interrupts have two types, Hardware interrupt and Software interrupt. The hardware interrupt occurs by the interrupt request signal from peripheral circuits. On the other hand, the software interrupt occurs by executing a dedicated instruction.

Buck Convertor-



- A buck converter (step-down converter) is a DC-to-DC power converter which steps down voltage from its input (supply) to its output (load).
- It is a class of switched-mode power supply (SMPS) typically containing at least two semiconductors (a diode and a transistor) and at least one energy storage element, a capacitor, inductor, or the two in combination.
- To reduce voltage ripple, filters made of capacitors are normally added to such a converter's output and input. It is called a buck converter because the voltage across the inductor "bucks" or opposes the supply voltage.

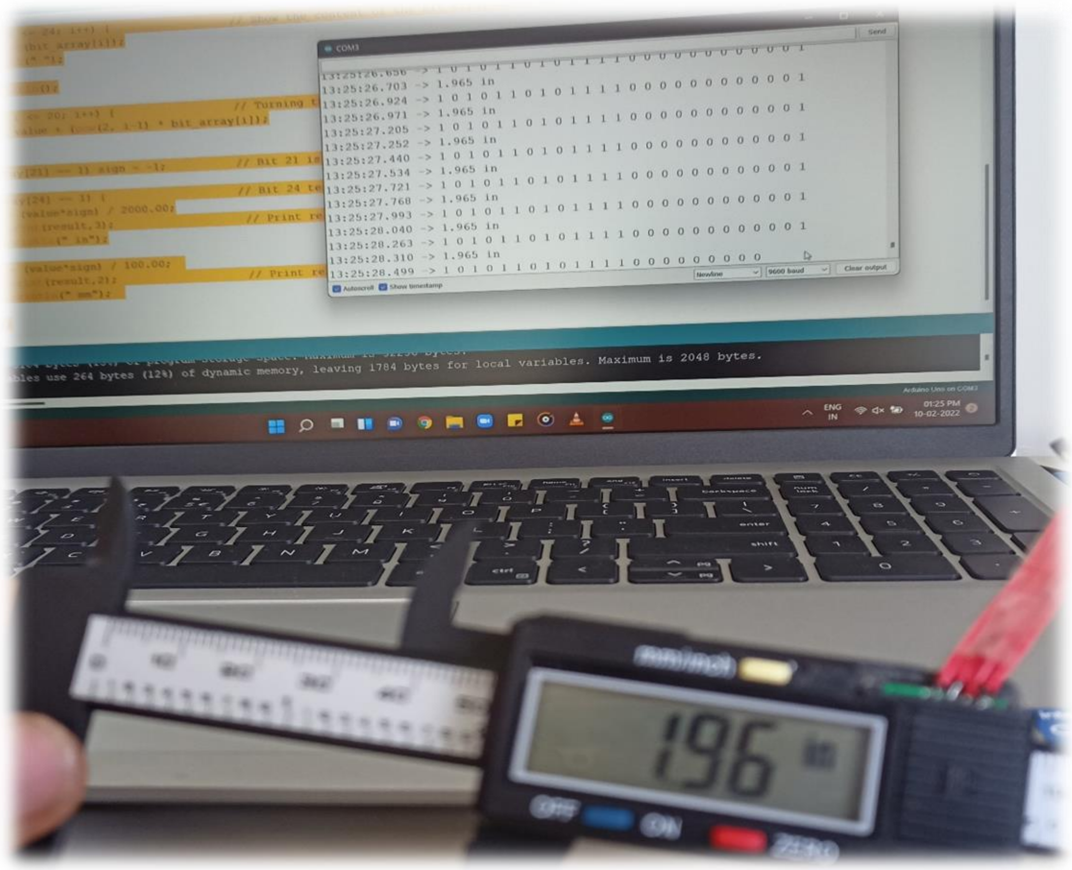
PROJECT: Interfacing the vernier caliper with Microcontroller and DRO {Digital Read Out} by using the Microcontroller.

Basically

The Digital Calliper is a precision instrument that can be used to measure internal and external distances extremely accurately. Data are transmitted by means of data line & clock line**. The calliper uses a 1.5V logic level.** We used Microcontroller to read the calliper measurement 1st on the serial monitor and then on the OLED display.

Finalized project view

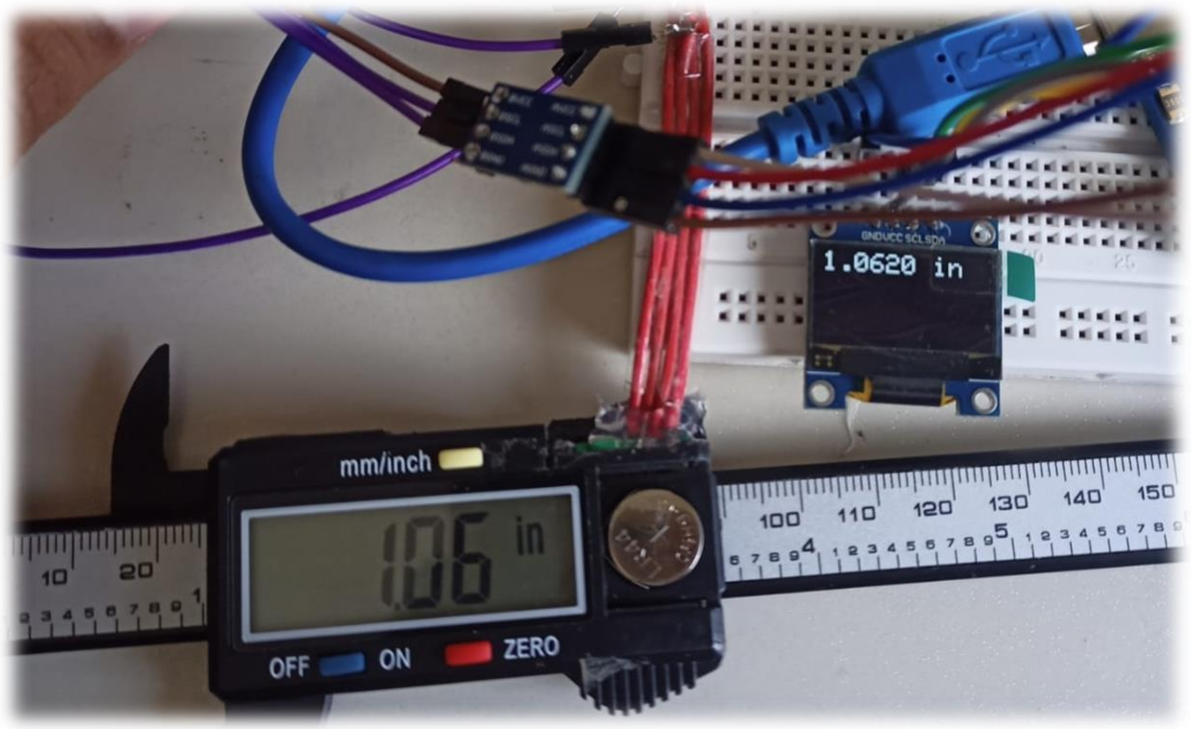
Getting caliper data on serial monitor

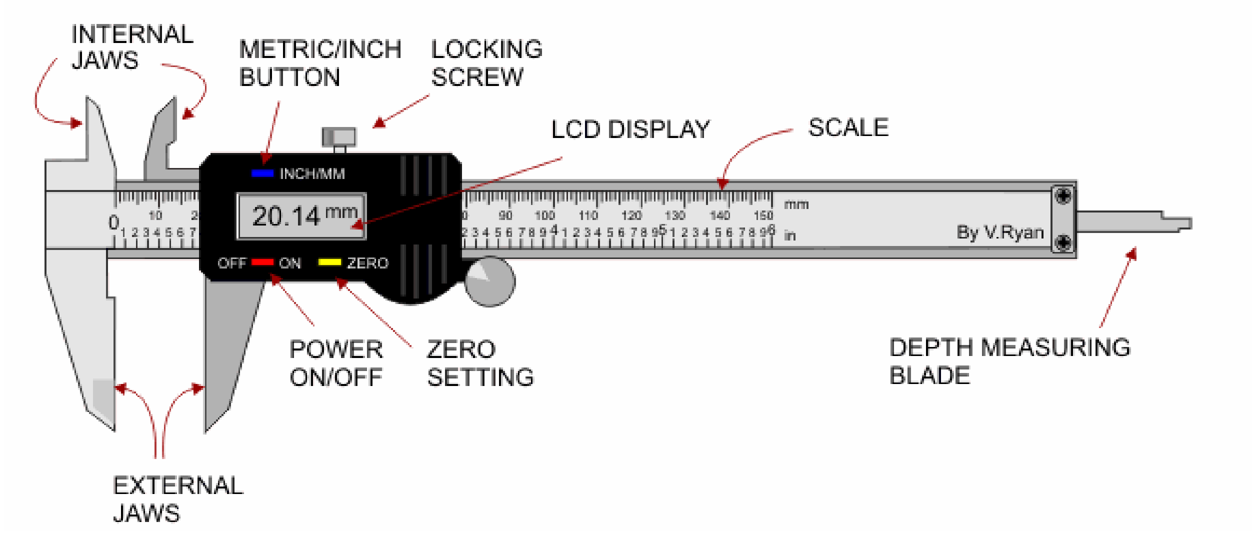


```
13:25:26.703 -> 1.965 in
13:25:26.924 -> 1 0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1
13:25:26.971 -> 1.965 in
13:25:27.205 -> 1 0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1
13:25:27.252 -> 1.965 in
13:25:27.440 -> 1 0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1
13:25:27.534 -> 1.965 in
13:25:27.721 -> 1 0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1
13:25:27.768 -> 1.965 in
13:25:27.993 -> 1 0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1
13:25:28.040 -> 1.965 in
13:25:28.263 -> 1 0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1
13:25:28.310 -> 1.965 in
13:25:28.499 -> 1 0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0
```

```
...les use 264 bytes (12%) of dynamic memory, leaving 1784 bytes for local variables. Maximum is 2048 bytes.
```

👉 Getting caliper *data* on OLED screen.



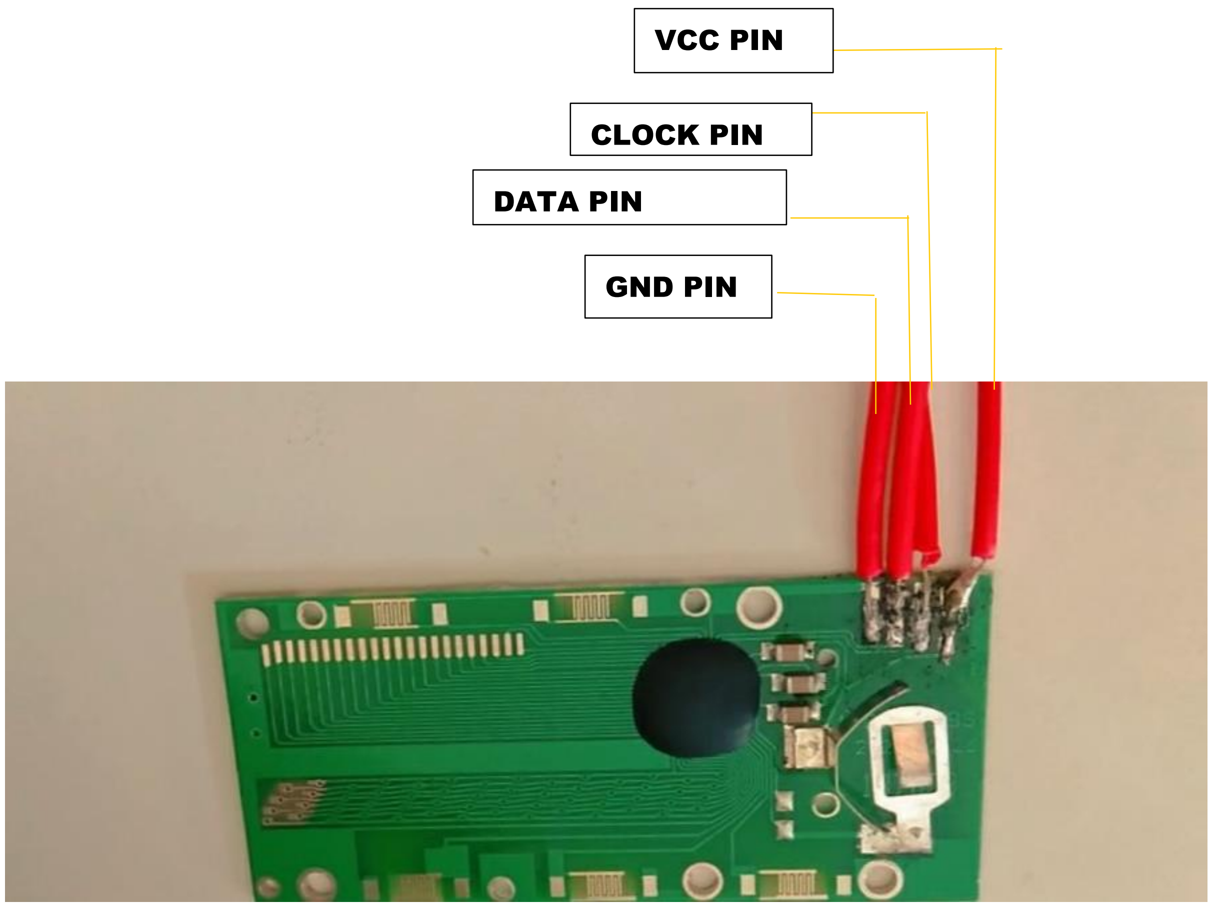


- DIGITAL VERNIER CALIPER is a very important tool in industries. It has wide range of application.

- On the in-built display of digital caliper we get only 1 or 2 bit precision { i.e. after decimal point only two digits} but by our project we can get up to 4 bit precision on both serial monitor and on OLED screen.

Soldering for getting data from digital caliper

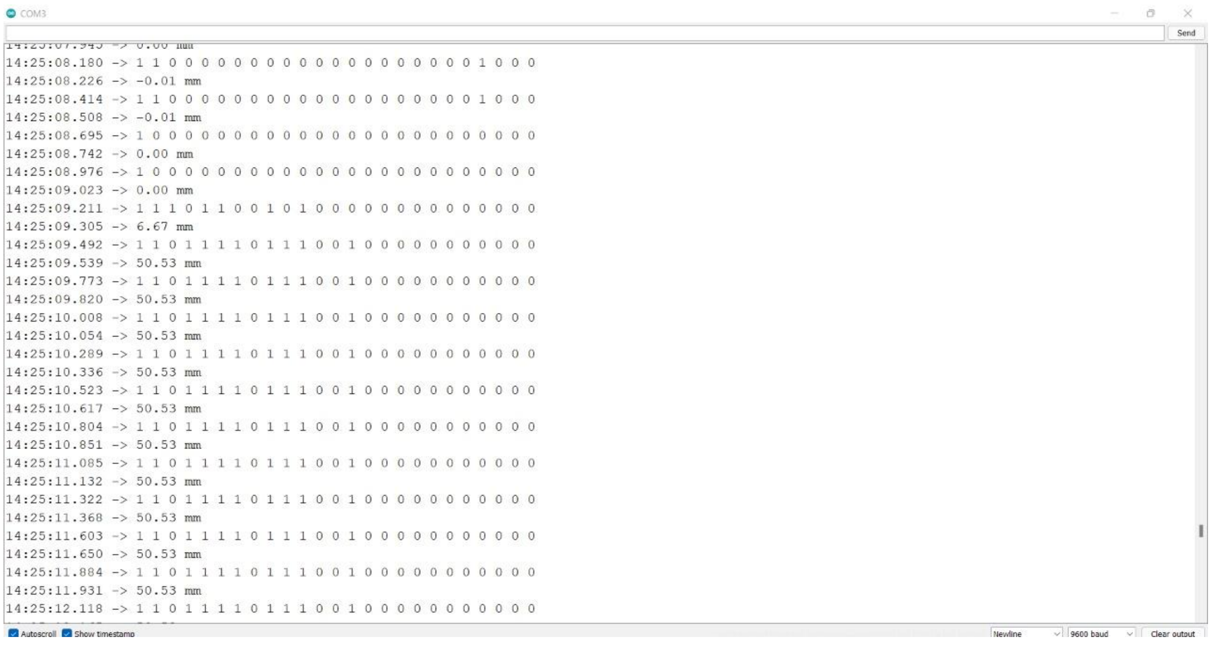
The PCB inside the digital caliper consist of a Microchip protected by black blob, some SMD resistors & capacitor. And 4 ports for digital readout {DRO} as shown below



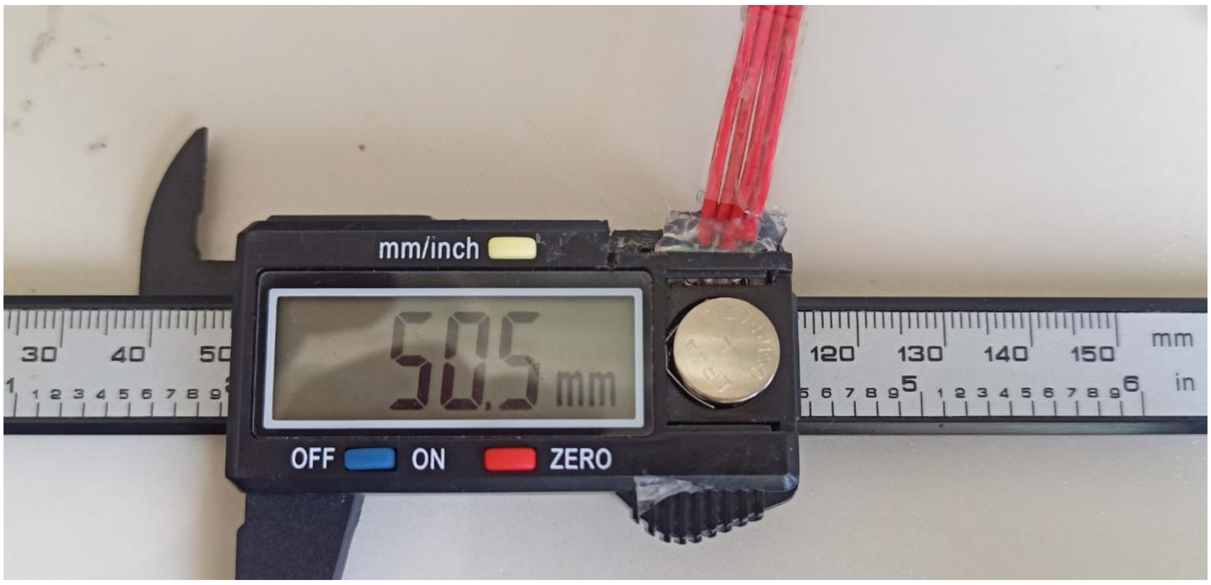
- As mentioned above we have learned I2C communication protocol. It requires 4 pin .
- So we soldered 4 wires at respective pins very carefully and then we connected that pins to the logic level shifter and ultimately to micro controller.

MAIN OUTPOUT ON SERIAL COM-

Proper 24-bit output from caliper and the code converts it into decimal format-a



```
14:25:08.180 -> 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
14:25:08.226 -> -0.01 mm
14:25:08.414 -> 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
14:25:08.508 -> -0.01 mm
14:25:08.695 -> 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
14:25:08.742 -> 0.00 mm
14:25:08.976 -> 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
14:25:09.023 -> 0.00 mm
14:25:09.211 -> 1 1 1 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
14:25:09.305 -> 6.67 mm
14:25:09.492 -> 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0
14:25:09.539 -> 50.53 mm
14:25:09.773 -> 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0
14:25:09.820 -> 50.53 mm
14:25:10.008 -> 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0
14:25:10.054 -> 50.53 mm
14:25:10.289 -> 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0
14:25:10.336 -> 50.53 mm
14:25:10.523 -> 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0
14:25:10.617 -> 50.53 mm
14:25:10.804 -> 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0
14:25:10.851 -> 50.53 mm
14:25:11.085 -> 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0
14:25:11.132 -> 50.53 mm
14:25:11.322 -> 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0
14:25:11.368 -> 50.53 mm
14:25:11.603 -> 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0
14:25:11.650 -> 50.53 mm
14:25:11.884 -> 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0
14:25:11.931 -> 50.53 mm
14:25:12.118 -> 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0
```



- **We can see here the caliper's lcd display is showing 50.5 mm only but our serial monitor is showing 50.53mm**

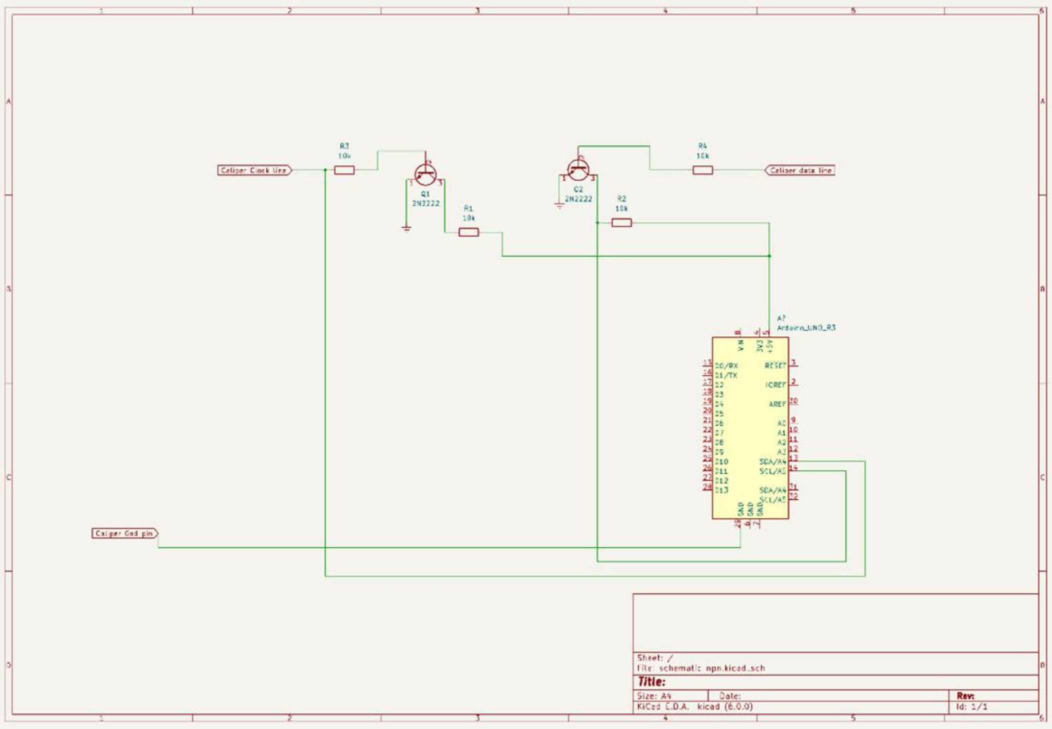
Bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Note	Start Bit	Bit 1 (LSB)	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8	Bit 9	Bit 10	Bit 11	Bit 12	Bit 13	Bit 14	Bit 15	Bit 16	Bit 17	Bit 18	Bit 19 (MSB)	Sign Bit 0->+ 1->-	-	-	In / mm 0 -> mm 1 -> In

REFERENCE

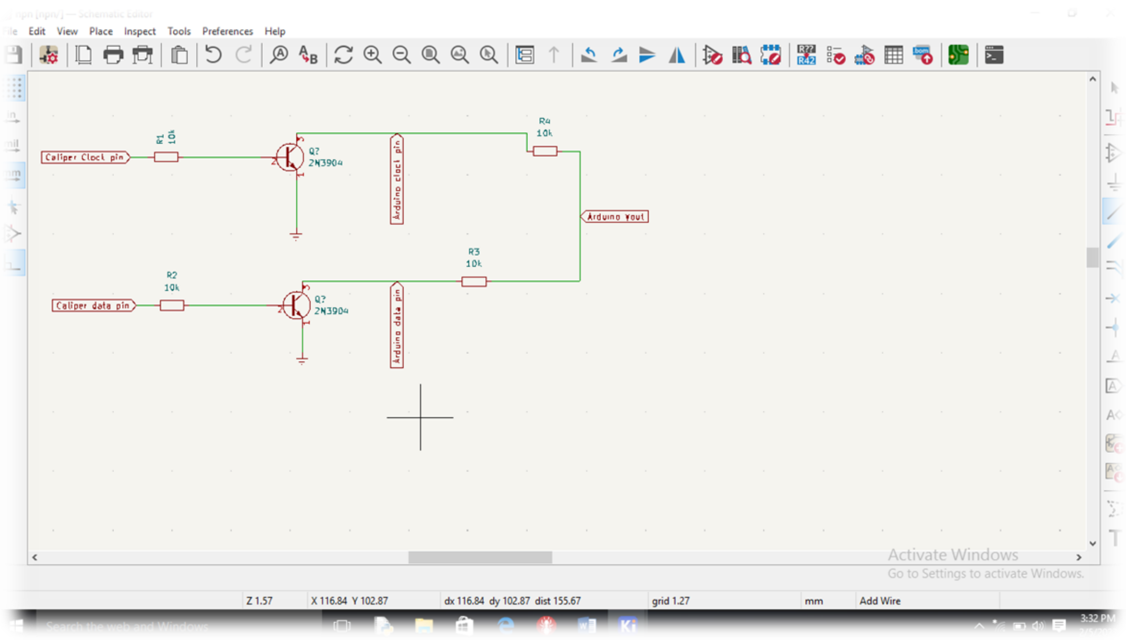
Some basic information about our project: -

- Caliper's data & clock signals are at 1.5V and need to be level shifted at microcontroller's VCC, so for that we need a comparator chip.
- When caliper >0.75 (i.e half the caliper voltage), then the comparator shifts to VCC.
- When caliper <0.75, comparator shifts to 0V
- MCU reads one packet bit whenever clock pin goes from VCC to 0(high to low)
- We have to shift 1.5V to 5V, as caliper runs on 1.5V & arduino on 5V, there needs to be a way to handle the difference in signal levels.
    1. Power the digital caliper using 3.3V output from arduino.
    2. Use transistor 2N2222 or BC548 to raise the logic level of clock & data output from caliper.
    3. Use logic level shifter (converts high to low signal or vice versa)
- A bi-directional Level Converter- 2 Channel is a small device that safely steps down 5V signals to 3.3V & steps up 3.3V to 5V at the same time.

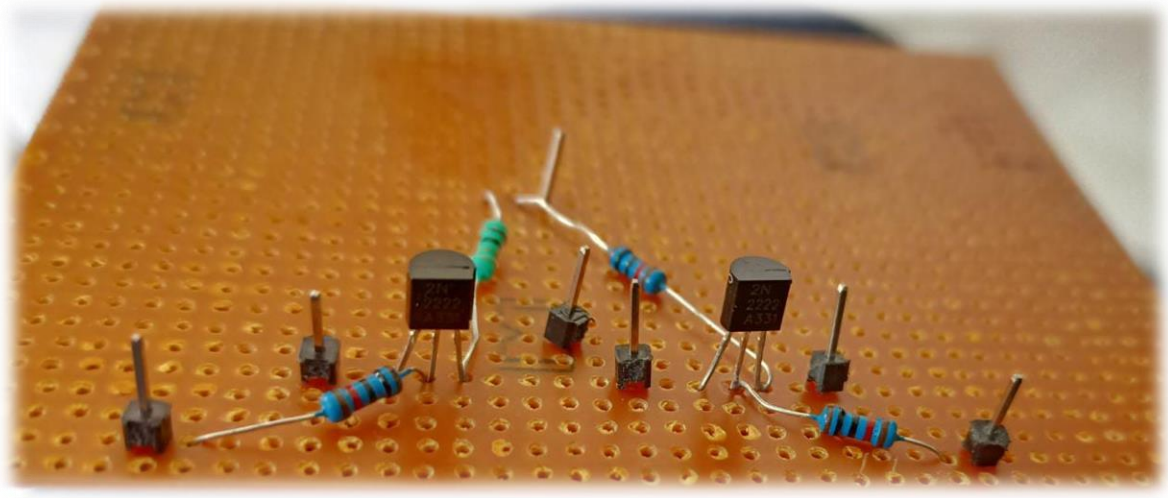
- <u>So we implemented logic level shifter on KI CAD.</u>



Schematic Designing of Logic Level Shifter using NPN Transistor-

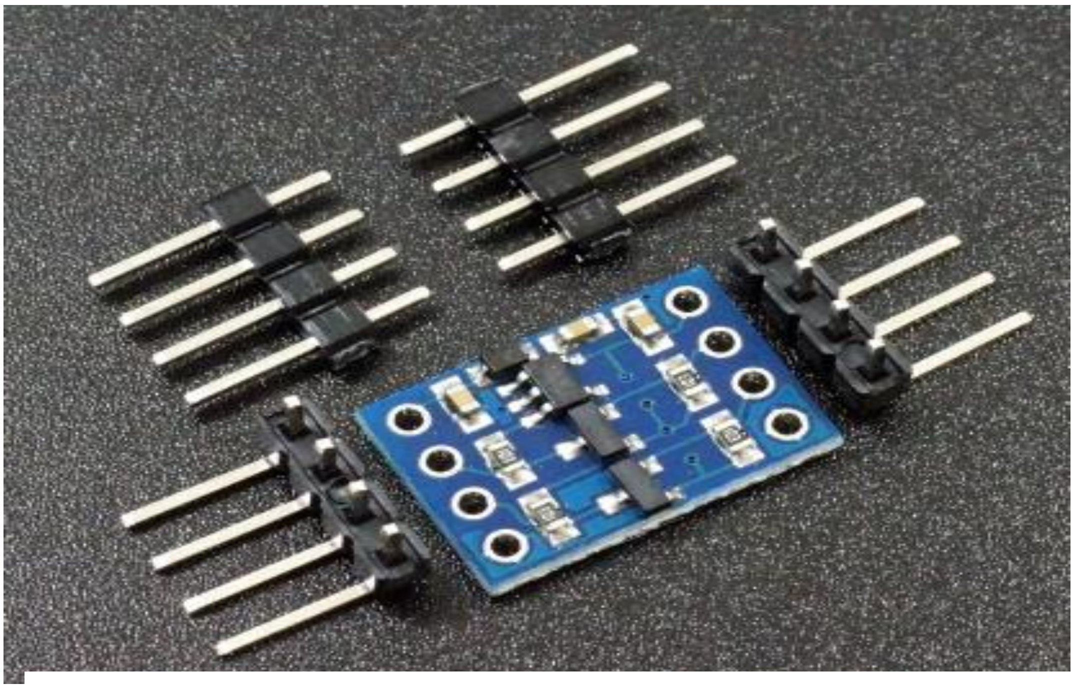


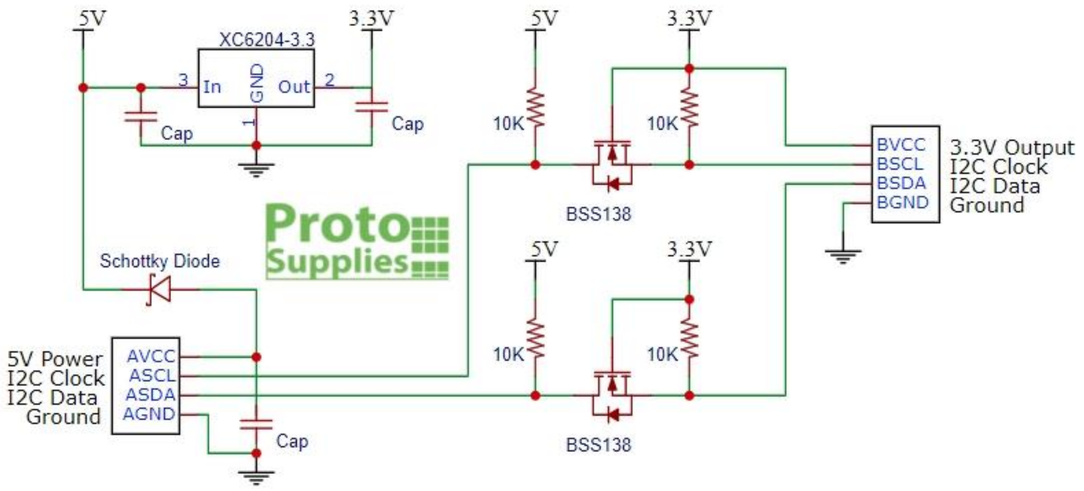
Practical Implementation on Dot Board-



We tried this own made circuit but it was not so accurate and wasn't working properly, so we have used LOGIC LEVEL SHIFTER instead. We have used this level shifter, as this logic converts high to low signal to low to high.

I2C Logic Level Shifter



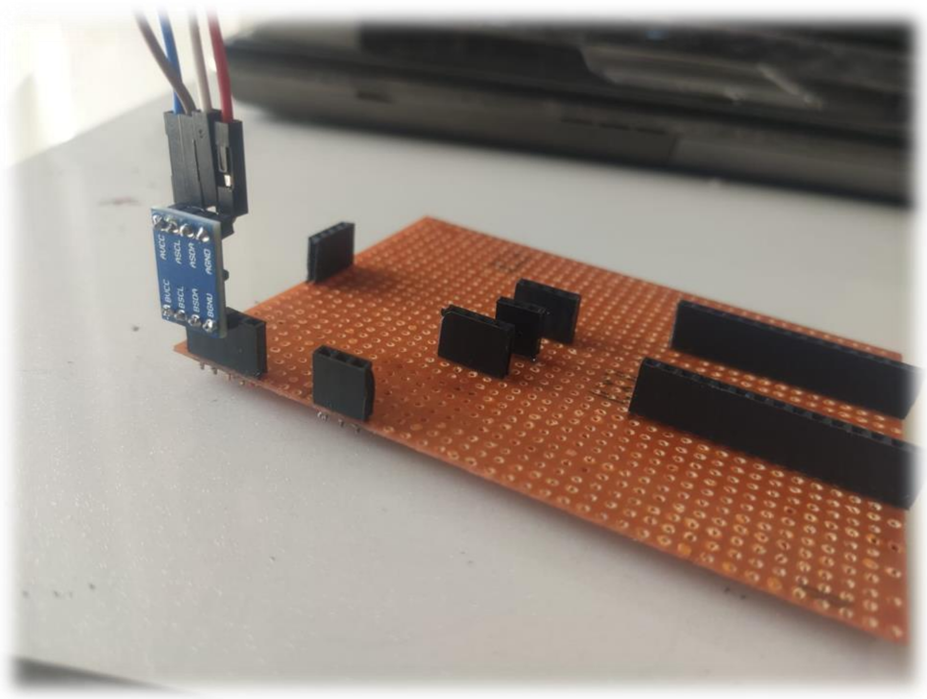


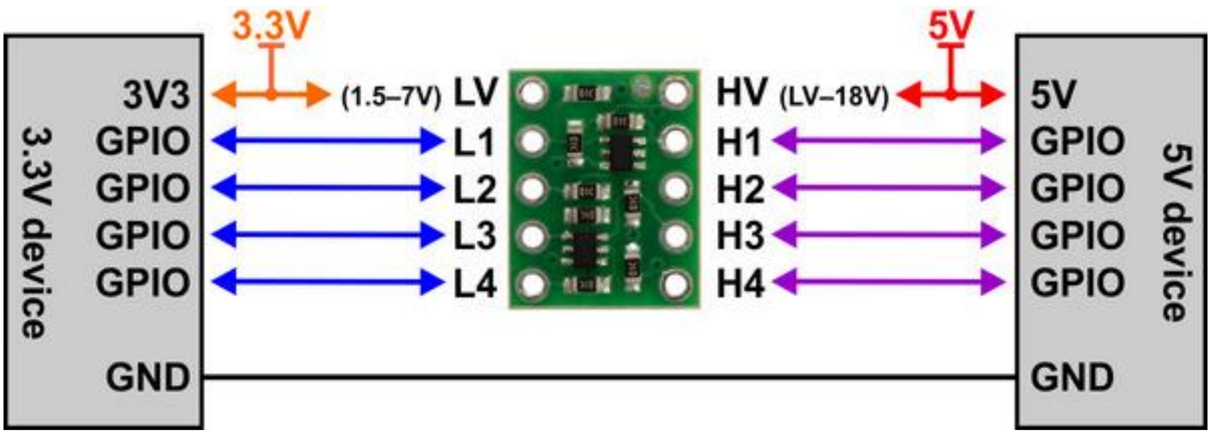
I2C Logic Level Converter with Regulator

Observation about logic level shifter.

Output voltage Range: 1.8V ~ 6.0V (XC6204) our module has this voltage regulator IC so if we common both ground then, excess voltage is supplied to the caliper and calipers LCD display all bits are high.
So by using this IC this issue can be resolved 0.9V ~ 1.75V (XC6205)

REFERENCE



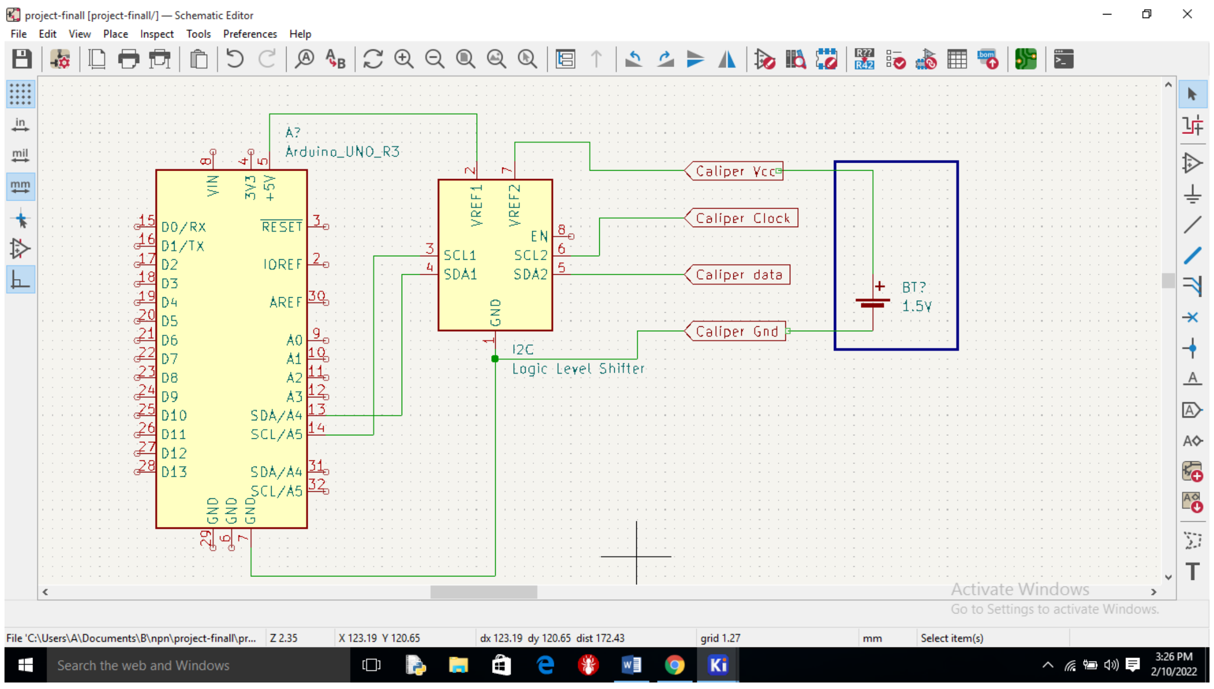


- The I2C Logic Level Converter with Regulator Module is designed to interface between 5V and 3.3V devices using the I2C bus.
- FEATURES OF LOGIC LEVEL CONVERTER –

1. 2 bi-directional channels can convert up to 2 logic signals, typically used for I2C
2. Can convert between 3.3V and 5V logic circuits
3. Utilizes N- MOSFET transistors for level translation
4. Includes on-board 3.3V regulator to power 3.3V logic
5. Operates from 5V power

- Module Operation
  1. The A side of the module is the 5V side while the B side of the module is the 3.3V side.
  2. The 5V logic SCL line connects to the ASCL pin and and the 3.3V SCL line connects to the BSCL pin.
  3. Similarly the 5V logic SDA line connects to the ASDA pin and the 3.3V SDA line connects to the BSDA pin.
  4. There are two Ground pins on the board as well. At least one should be connected to the 5V circuit. The other ground can be used to provide a ground to the 3.3V peripheral if needed.
- Module Connections-
  AVCC = 5V input
  BVCC = 3.3V output
  GND = Ground (x2). At least one needs to connect to 5V ground.
  ASCL / BSCL = I2C Clock channel
  ASDA / BSDA = I2C Data channel

Schematics of final project with working code-



Digital Caliper reading with Arduino on OLED display code.

```
        #include<Wire.h>

        #include<Adafruit_GFX.h>

        #include<Adafruit_SSD1306.h>

        #define screen_width 128

        #define screen_height 64

        Adafruit_SSD1306 display(screen_width,screen_height);


        int bit_array[25];        // For storing the data bit. bit_array[0] = data bit 1 (LSB),
bit_array[23] = data bit 24 (MSB).

        unsigned long time_now;   // For storing the time when the clock signal is
changed from HIGH to LOW (falling edge trigger of data output).


        int CLOCK_PIN = 2;
int DATA_PIN = 3;


void setup() {
  //Serial.begin(9600);
  pinMode(CLOCK_PIN, INPUT);


  pinMode(DATA_PIN, INPUT);
  display.begin(SSD1306_SWITCHCAPVCC, 0X3C);
}




void loop() {
   display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0,0);
  while (digitalRead(CLOCK_PIN) == LOW) {}  // If clock is LOW wait until it
turns to HIGH
  time_now = micros();
  while (digitalRead(CLOCK_PIN) == HIGH) {} // Wait for the end of the HIGH
pulse
  if ((micros() - time_now) > 500) {        // If the HIGH pulse was longer than 500
micros we are at the start of a new bit sequence
    decode(); //decode the bit sequence
  }
}


void decode() {
  int sign = 1;
```

```
        int i = 0;

        float value = 0.0;

        float result = 0.0;


        bit_array[i] = digitalRead(DATA_PIN);     // Store the 1st bit (start bit) which
is always 1.

        while (digitalRead(CLOCK_PIN) == HIGH) {};


        for (i = 1; i <= 24; i++) {

          while (digitalRead(CLOCK_PIN) == LOW) { } // Wait until clock returns to
HIGH

          bit_array[i] = digitalRead(DATA_PIN);

          while (digitalRead(CLOCK_PIN) == HIGH) {} // Wait until clock returns to
LOW

        }


        for (i = 0; i <= 24; i++) {          // Show the content of the bit array. This is
for verification only.

          //Serial.print(bit_array[i]);

          //Serial.print(" ");

        }

         // Serial.println();


        for (i = 1; i <= 20; i++) {          // Turning the value in the bit array from
binary to decimal.

            value = value + (pow(2, i-1) * bit_array[i]);

        }


        if (bit_array[21] == 1) sign = -1;       // Bit 21 is the sign bit. 0 -> +, 1 => -
        if (bit_array[24] == 1) {             // Bit 24 tells the measuring unit (1 -> in, 0 ->
mm)
          result = (value*sign) / 2000.0000;
          //Serial.print(result,4);             // Print result with 3 decimals
         // Serial.println(" in");
          display.print(result,4);
          display.print(" in");
        display.setCursor(40,40);
        //display.print(millis());
        display.display();
        }else {
          result = (value*sign) / 100.0000;
        //  Serial.print(result,2);             // Print result with 2 decimals
          //Serial.println(" mm");
          display.print(result,4);
          display.print(" mm");
        display.setCursor(40,40);
        //display.print(millis());
        display.display();


        }
        delay(5);
       }
```

OBSERVATION-

- We have tried this own made circuit but it was not so accurate and wasn't working properly, so we have used LOGIC LEVEL SHIFTER instead. We have used this level shifter, as this logic converts high to low signal to low to high.
- There are 4 pins VCC, clock, data, ground.
- Level shifter module can be used to shift 1.5V logic to arduino 5V
- Caliper is powered from cell, set to 1.5V

- On both sides, 4 level shifter pins are used.
- Returned data provide 24 bit resolution.