FINAL PROJECT REPORT

WEATHER STATION WITH BLUETOOTH AND WIFI MODULES

-BY

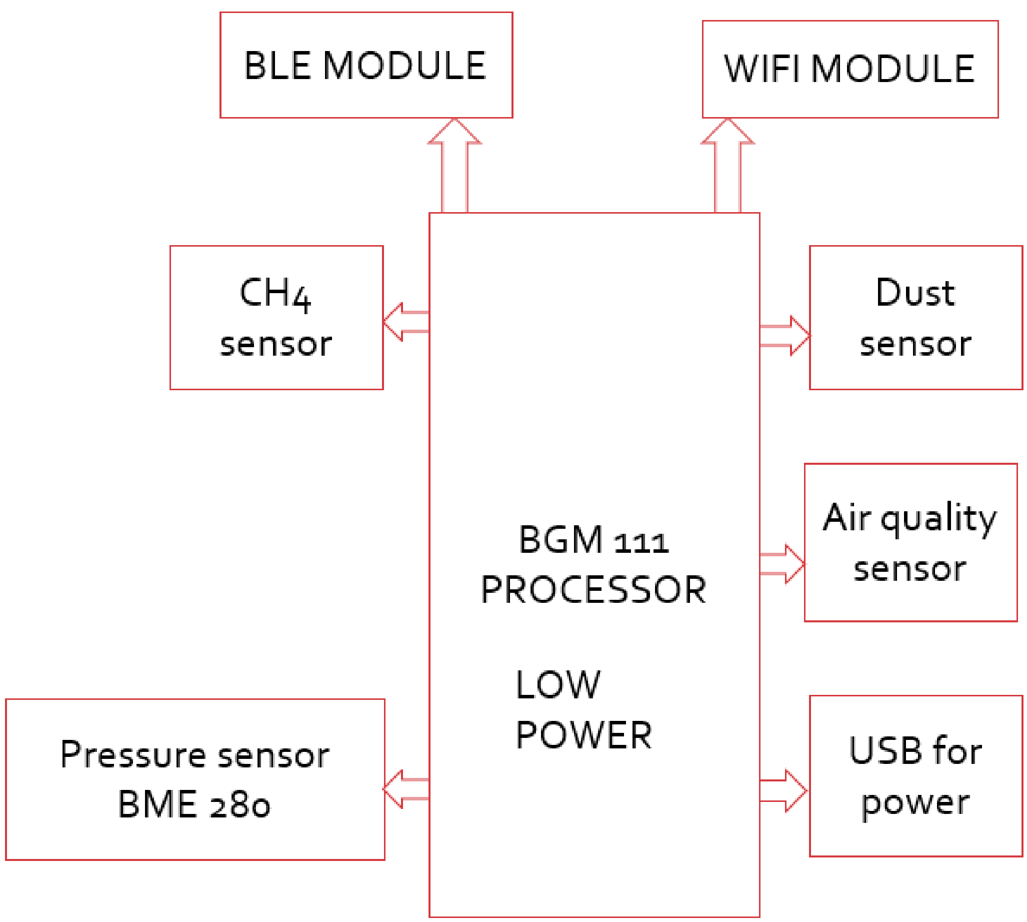
SANIKA DONGRE

SAHANA SADAGOPAN

INDEX

TABLE OF CONTENT	PAGE NO.
BLOCK DIAGRAM	3
COMPONENTS	4
SCHEMATIC OF THE CIRCUIT	5-9
PCB 4-layer Routing	10
BOARD PLACEMENT AND CONNECTIONS	11-12
WORKING	13
TESTING AND RESULTS	14-19
LESSONS LEARNT	20-21
CODES	22-27
FUTURE SCOPE, CONCLUSION & REFERENCES	28

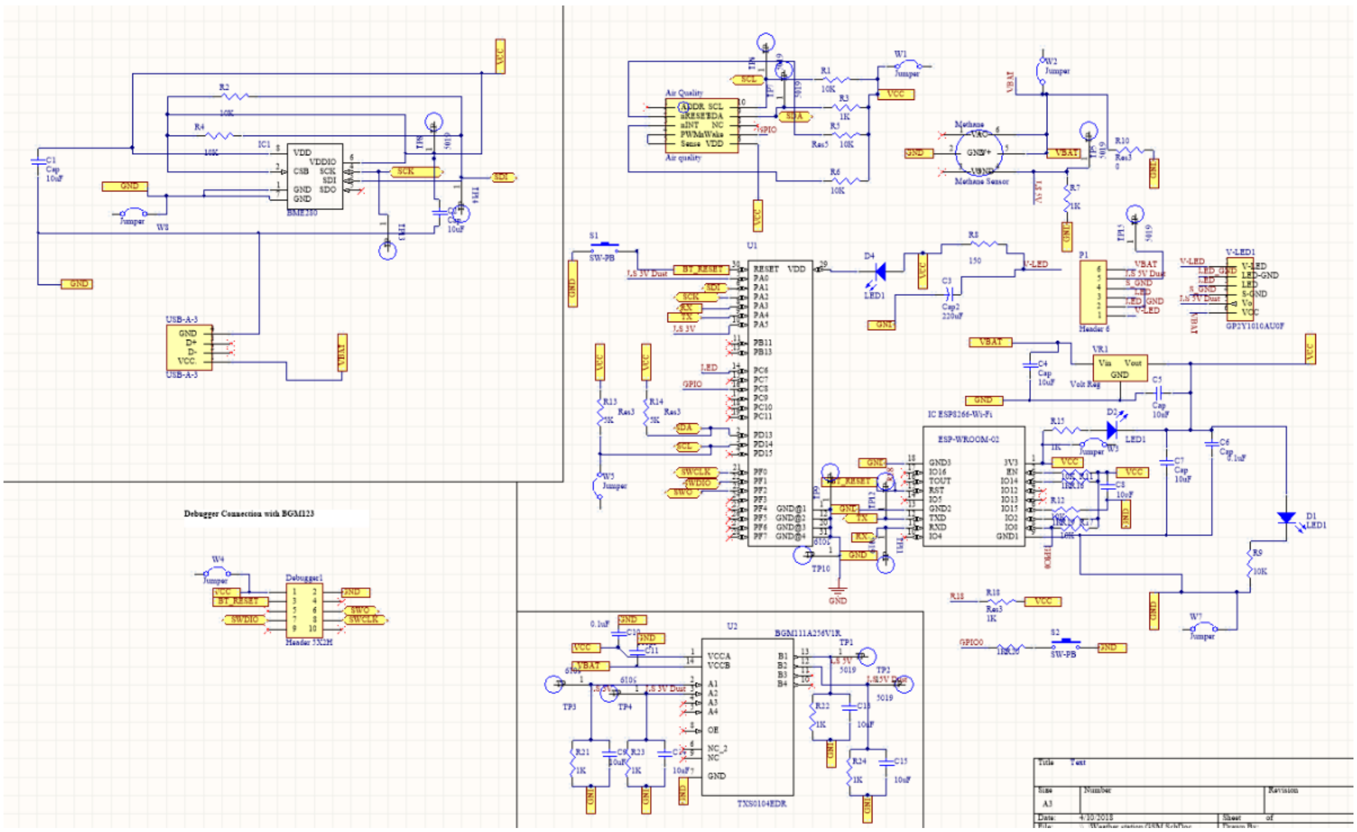
BLOCK DIAGRAM:



COMPONENTS:

- PROCESSOR: BGM111
- MODULES: BLUETOOTH (BLUEGECKO) AND WIFI MODULE (WROOM ESP 8266)
- FEMALE CHARGING PIN USB A-3.0
- LEVEL SHIFTER IC
- VOLTAGE REGULATOR
- RESISTORS (10K (13), 1K (7), 150 (1), 0(1))
- CAPACITORS (10uF- (12), 1uF- (1), 220uF-(1))
- LED INDICATORS
- PUSH BUTTON SWITCHES
- TEST POINTS (LOOP TYPES) (15)
- PRESSURE SENSOR BME 280
- AIR QUALITY SENSOR (CCS811)
- METHANE SENSOR (MQ2)
- DUST SENSOR (GP2Y1010AU0F)
- HEADERS FOR DUST SENSOR
- HEADERS FOR MAKING CONNECTION for DEBUGGER REQUIRED FOR BGM111
- JUMPER WIRE HEADERS FOR MEASURING VOLTAGES

SCHEMATIC:



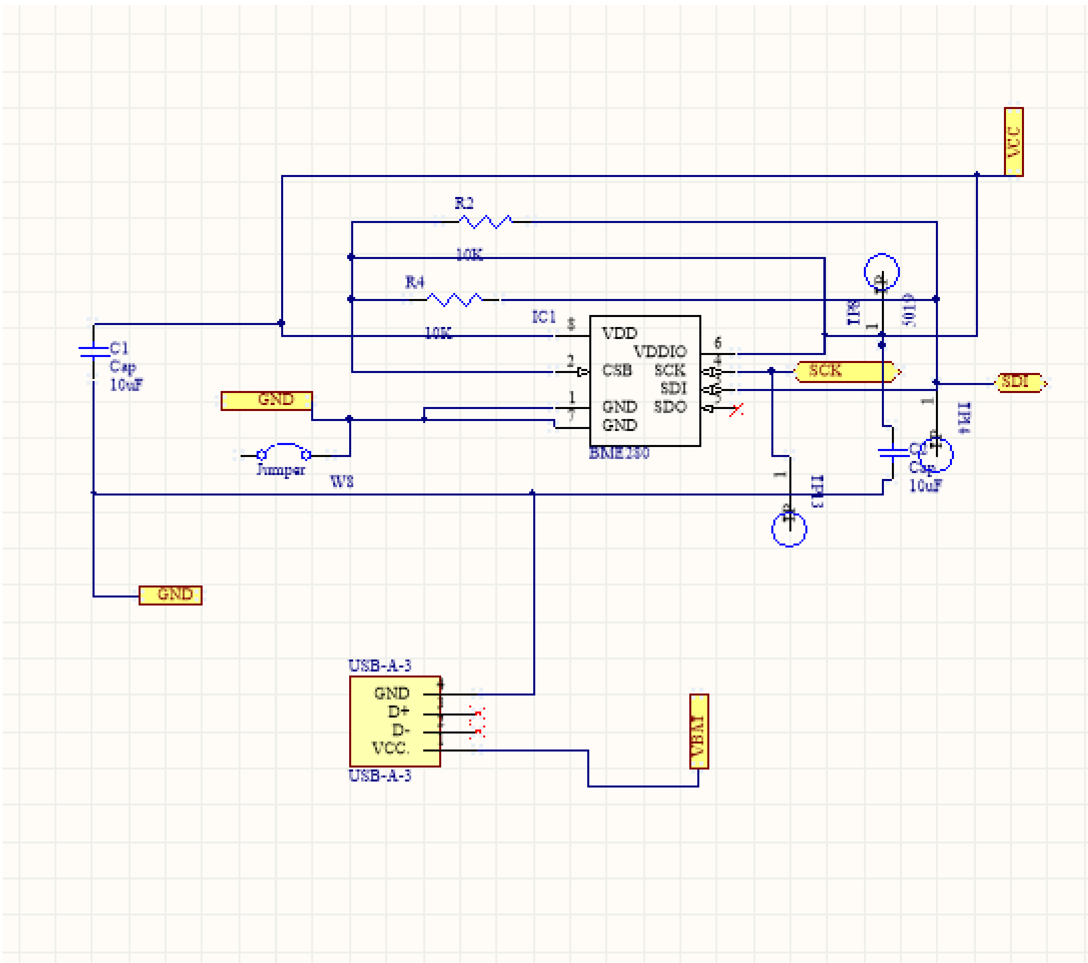
We have used Net labels and ports

Instead of making different small sheets and linking them, we have used a single large sheet (A3 size) [as taught in class]

ZOOMING IN DIFFIRENT AREAS OF SCHEMATIC AND ITS DESCRIPTION:

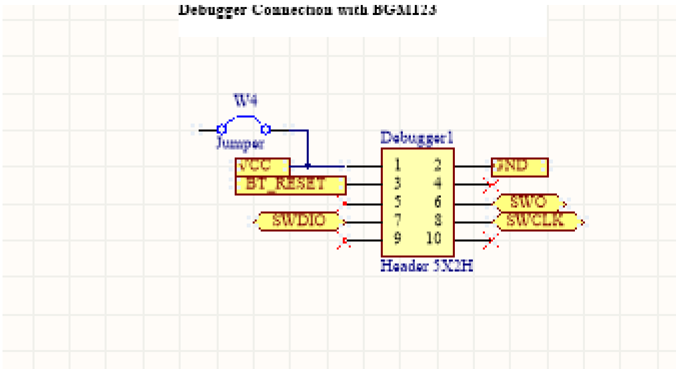
SCHEMATIC: PART 1: DESCRIPTION:

This is the pressure sensor BME 280 which is being connected to the USB and the BGM (using ports). The USB is used to provide 5v to the circuit.



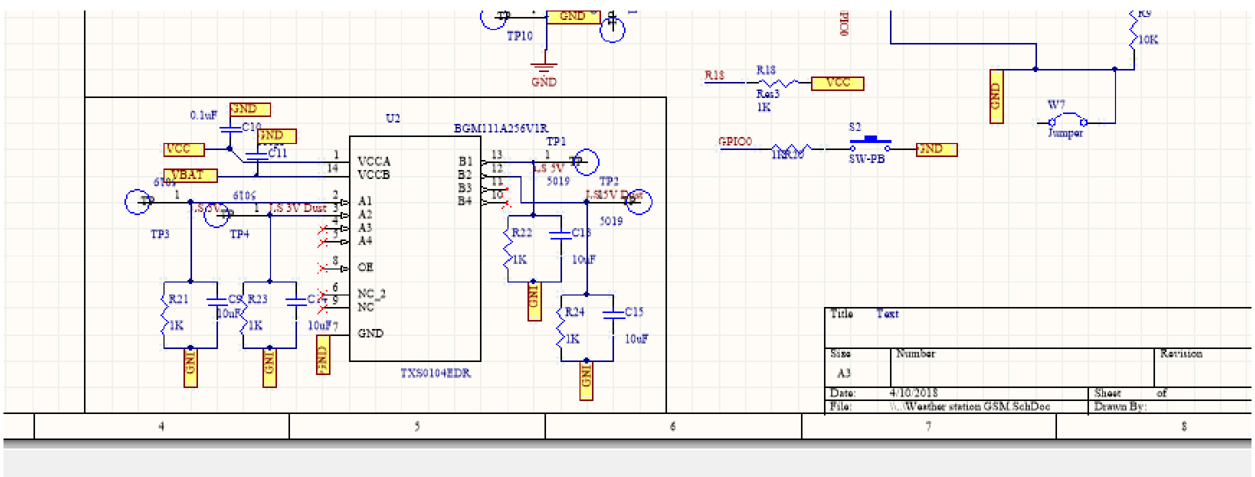
SCHEMATIC: PART 2: DESCRIPTION:

This is the debugger connection with the BGM113



SCHEMATIC: PART 3: DESCRIPTION:

This is the schematic for the level shifter IC used in the circuit because few components require 3.3 v while others require 5v for operation, so it will perform the voltage translation.



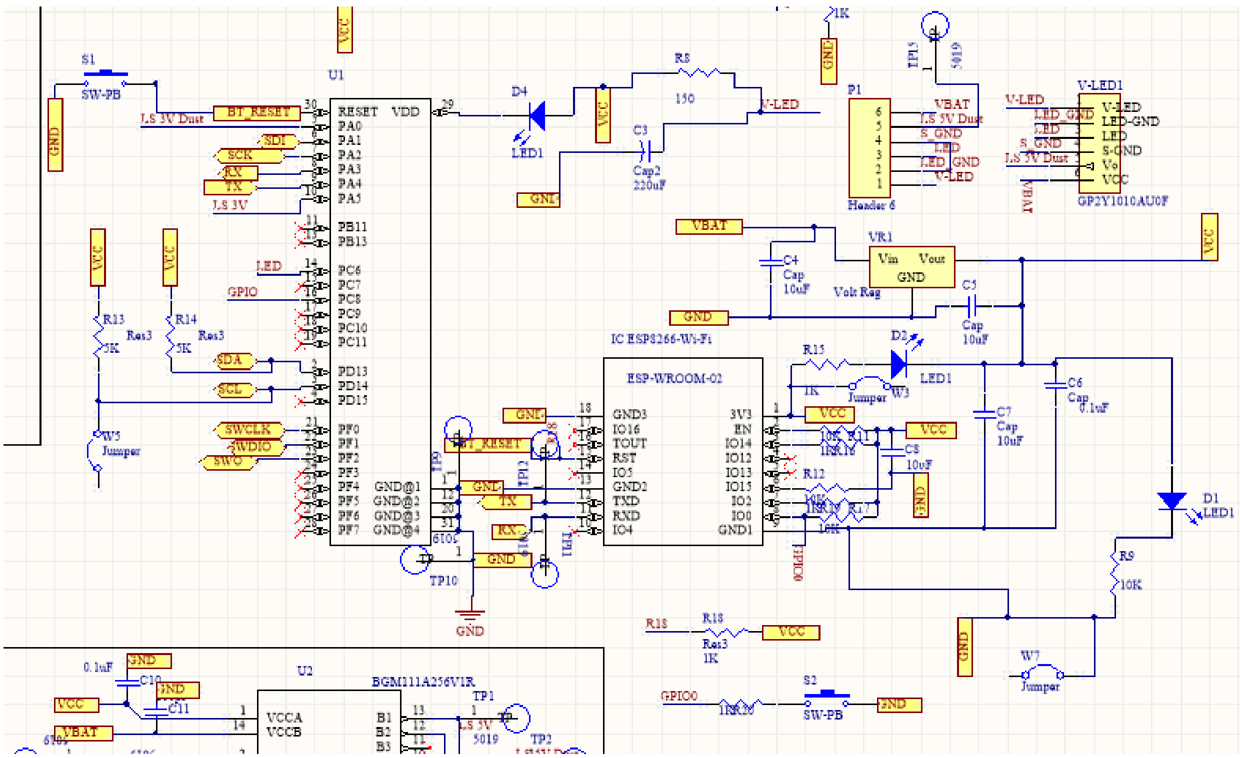
SCHEMATIC: PART 4: DESCRIPTION:

THE IC U1 in the schematic is the BGM111 processor.

The ESP WROOM 8266 is the WIFI module

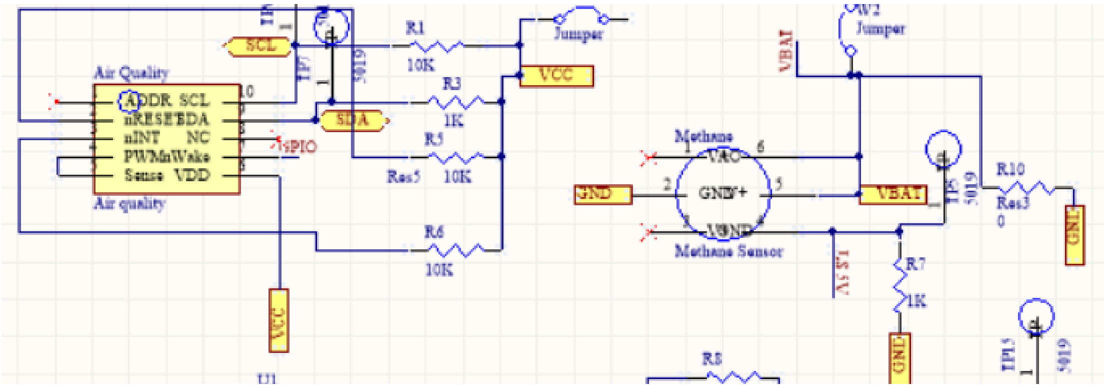
The header 6 is used for making connections to the dust sensor

The VR1 is the voltage regulator used to provide 3.3 volts to the BGM111 and ESP WROOM 8266. The VR1 gets an input voltage of 5v from the USB and converts it to 3.3v.



SCHEMATIC: PART 5: DESCRIPTION:

This is the Air quality sensor (left) and the methane sensor circuit (right) are being connected by referring to the datasheet. The methane sensor is connected to Analog compare pin of BGM111 and the Air quality sensor is connected to the I2C SDA and SCL lines of BGM111 using ports



ROUTING:

The mechanical layer on the left side is marked for the placement of the dust sensor which is neither through hole nor surface mount and is thus connected with the P1 header.

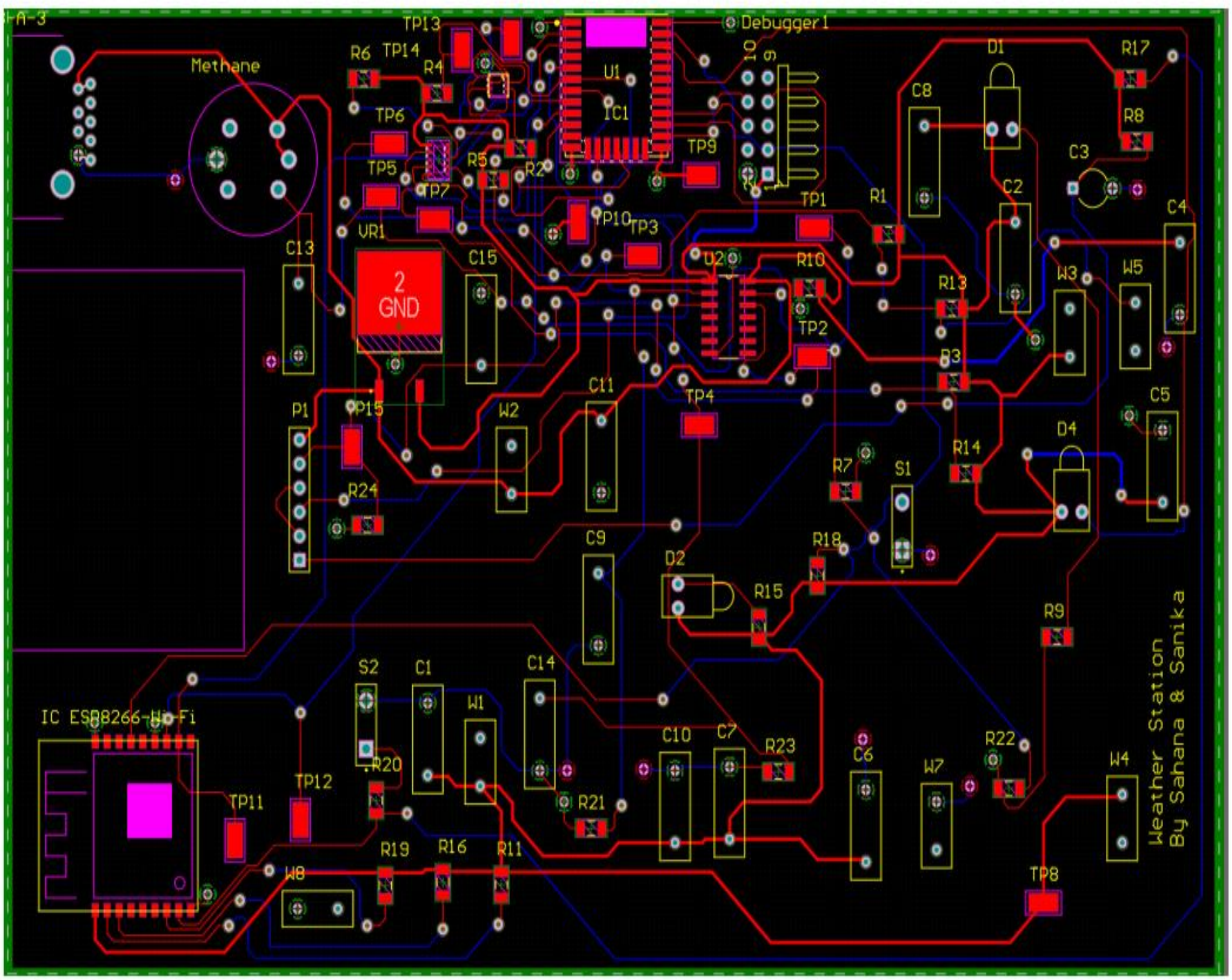
LAYER 1: PWR/SIG

LAYER 2: GND PLANE 1

LAYER 3: GND PLANE 2

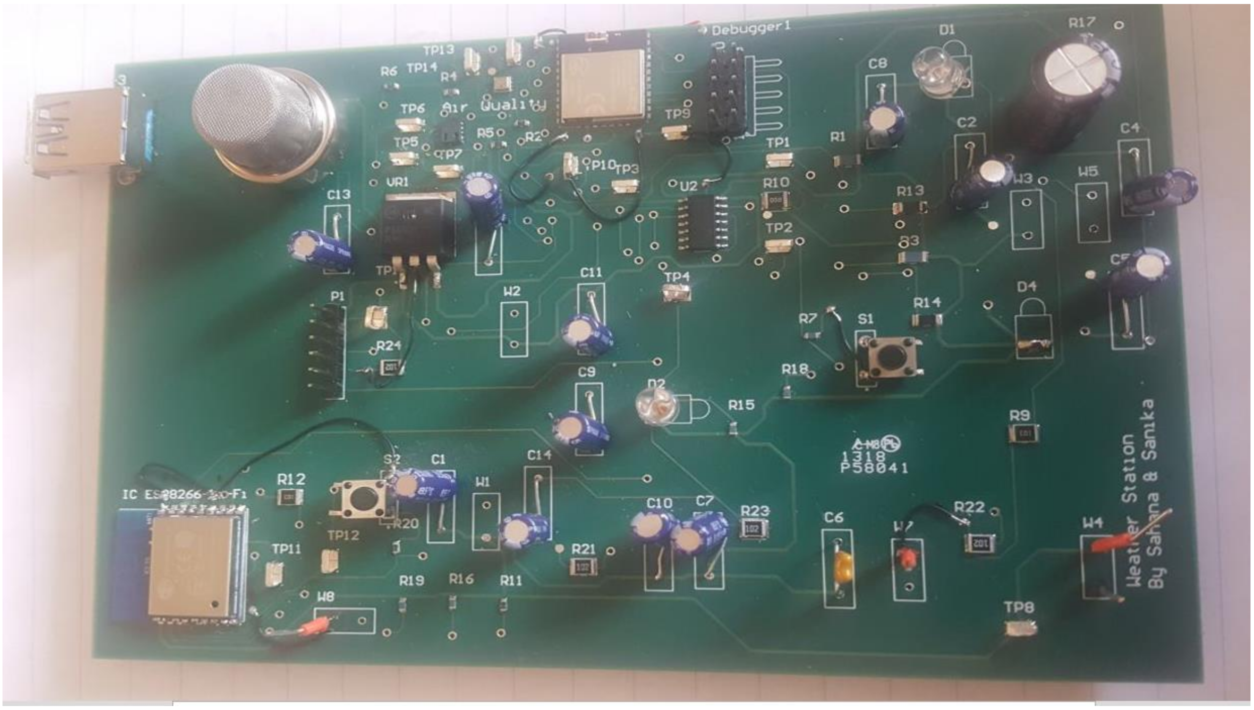
LAYER 4: PWR/SIG

Routing

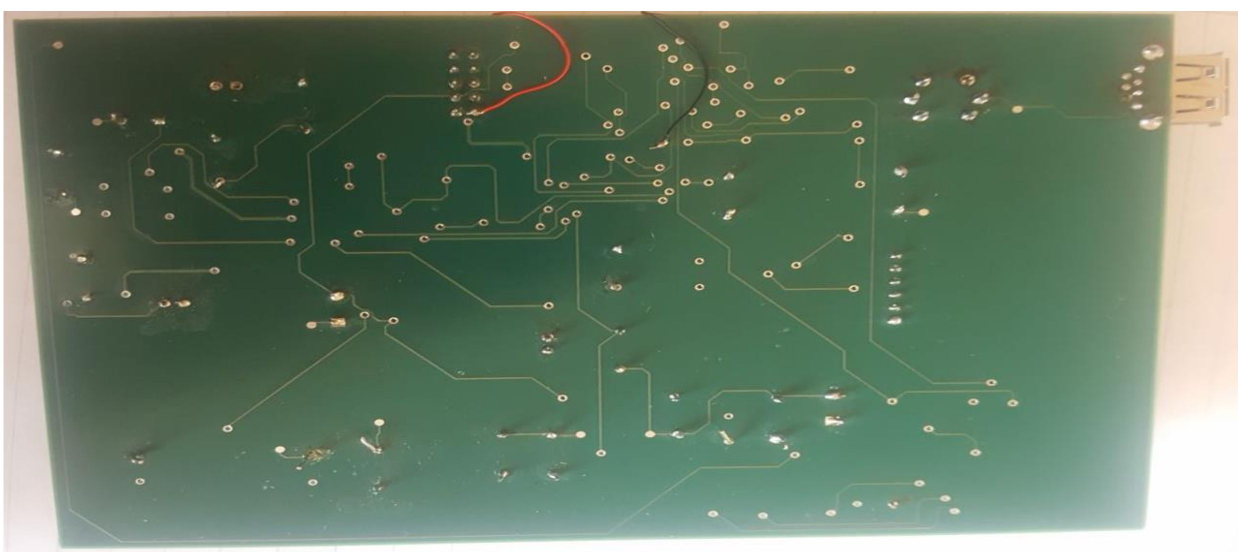


BOARD PLACEMENT

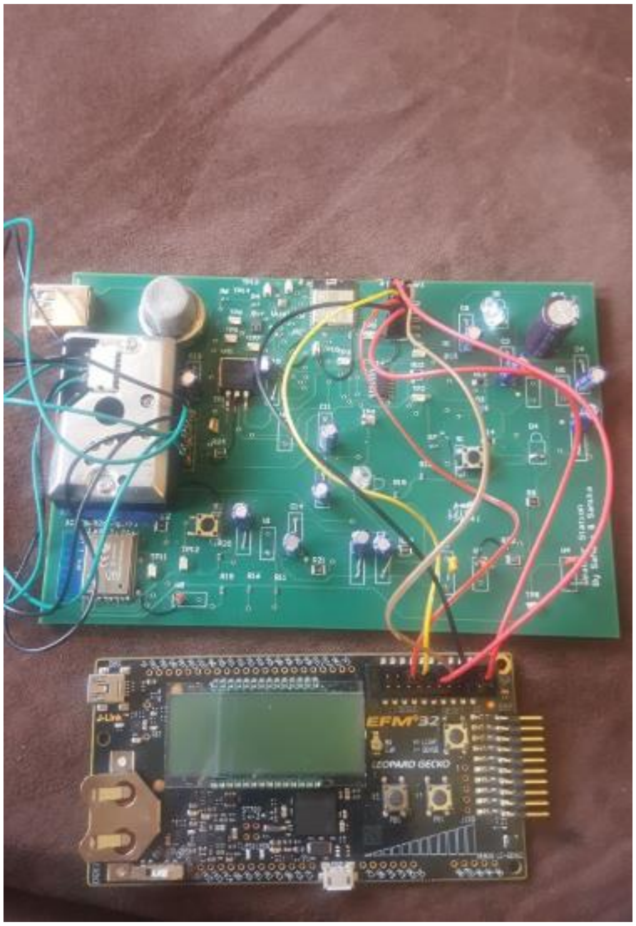
FRONT SIDE:



BACK SIDE:



CONNECTION OF THE PCB BOARD WITH THE DEVELOPMENT BOARD USING THE DEBUGGER HEADERS AND THE DUST SENSOR IS NEITHER THROUGH HOLE NOR SURFACE MOUNT SO IT IS CONNECTED USING HEADERS (PLACED ON THE LEFT SIDE OF BOARD – BELOW THE USB)



WORKING:

In this project, we are creating a prototype of a weather station that is used for detecting the quality of the air, the percentage of LPG, methane, CO2 in ppm, the amount of dust in the air and the pressure by integrating various sensors modules on a single PCB board. The data from the sensors is being given to respective pins (for eg: MQ2 sensor has an analog output so the data is given to the analog comparator pins of the blue-gecko) of the blue gecko. The BGM-121 module can process data and has an in built bluetooth antenna. Using Bluetooth low energy, the output of the sensor readings is sent over the phone. This data is also being sent on the server of a PC creating a website to display outputs by using the WiFi module. The sensors are interfaced using the following Bus protocols:

- Dust sensor: ADC
- Air Quality sensor: I2C
- Pressure sensor BME 280: SPI
- Methane sensor: ADC

The Characteristic impedance is: 10 ohms

TESTING CONDITIONS

- Input/output voltage of the sensors
- Input/output current rating of the sensors
- Compatibility of the sensors with the blue gecko
- Voltage drop across various test points
- Feasibility and the efficiency of the project idea
- LEDs to indicate the functioning of the sensors
- Measuring the load voltage at certain intervals.
- Testing the protocols used for interfacing sensors

TESTING AND RESULTS:

- The Voltage values and the current values expected across all sensors and the BGM and WIFI Module

Sensor/Component	Voltage Value	Current Value
MCU(BGM111)	3.5V	10mA(No Power saving usage)
Dust Sensor	5V	20mA
Air Quality	3.5V	54mA
Temperature and Humidity	3.3V	-
Voltage regulator	3.3V	-
Wi-Fi	3.3V	-
Level shifter	3.3V	100-150mA(Total current
Methane Sensor	5v	-

Voltages measured across various test points

Test Points	Voltage
Test point 1(output of level shifter)	4.97V
Test point 2(Dust Sensor input)	4.99V
Test point3(Input of level shifter to dust)	3.55
Test poin4(methane input to bgm111)	3.1v
Test point 5,Test point 9	GND
Test point 6(I2C BME)	I2C_SCL
Test point 7	I2C_SDA
Test point 8(Voltage to Wifi Module)	3.3V
Test point10	GND
Test point11	2.5V
Test point 12	Wi-Fi Pin
Test point 13	2.5V
Test point 14(analog pin methane)	3.3V
Test point 15	5v

OBTAINED DUST SENSOR READINGS:

Dust Sensor values are derived by comparing the Digital and analog values obtained on the pin and we obtained 0.05mg/m3 to 0.08mg/m3 in varied residential location

The calculations:

calcVoltage = voMeasured * (5 / 1024); // where 1024 is mapped to 5V

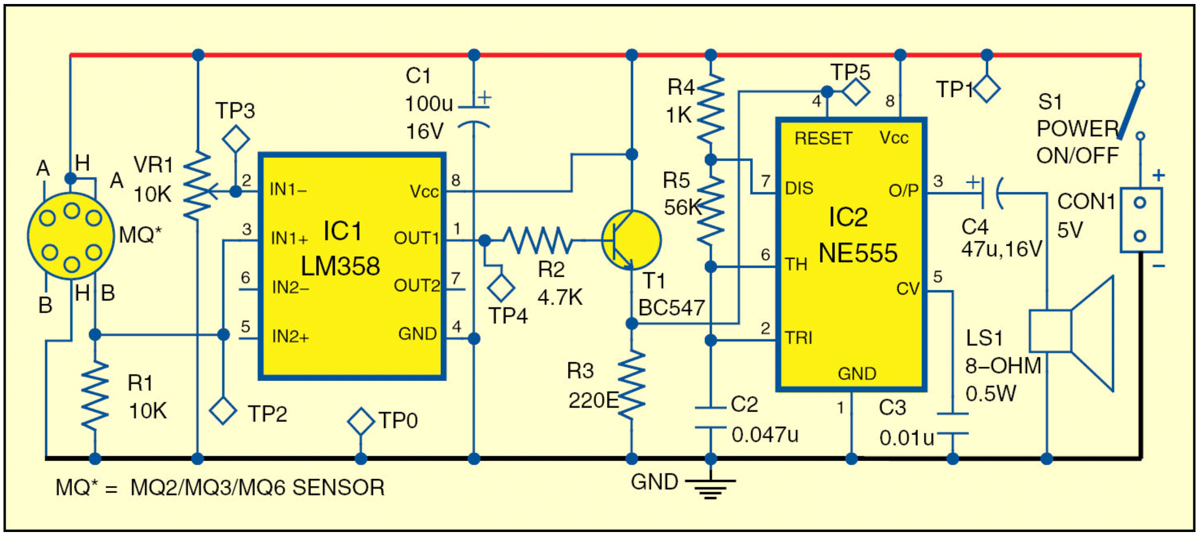
Dust Sensor = 0.17*calcVoltage-0.1 // Calculation in the data sheet

```
Raw Signal Value (0-1023): 300.00 - Voltage: 0.97 - Dust Density: 0.06
Raw Signal Value (0-1023): 283.00 - Voltage: 0.91 - Dust Density: 0.06
Raw Signal Value (0-1023): 280.00 - Voltage: 0.90 - Dust Density: 0.05
Raw Signal Value (0-1023): 298.00 - Voltage: 0.96 - Dust Density: 0.06
Raw Signal Value (0-1023): 288.00 - Voltage: 0.93 - Dust Density: 0.06
Raw Signal Value (0-1023): 279.00 - Voltage: 0.90 - Dust Density: 0.05
Raw Signal Value (0-1023): 291.00 - Voltage: 0.94 - Dust Density: 0.06
Raw Signal Value (0-1023): 292.00 - Voltage: 0.94 - Dust Density: 0.06
Raw Signal Value (0-1023): 282.00 - Voltage: 0.91 - Dust Density: 0.05
```

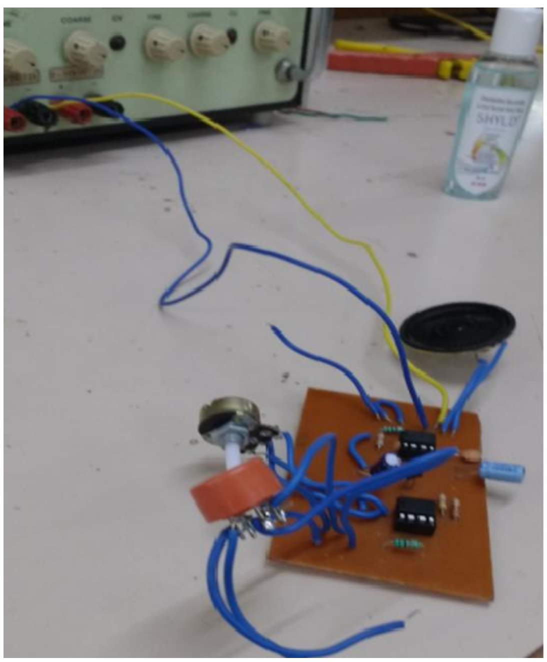
OBTAINED AIR QUALITY READINGS IN INDOOR ENVIRONMENT:

The values are read directly from the data register CO2 values between 600ppm to 700ppm

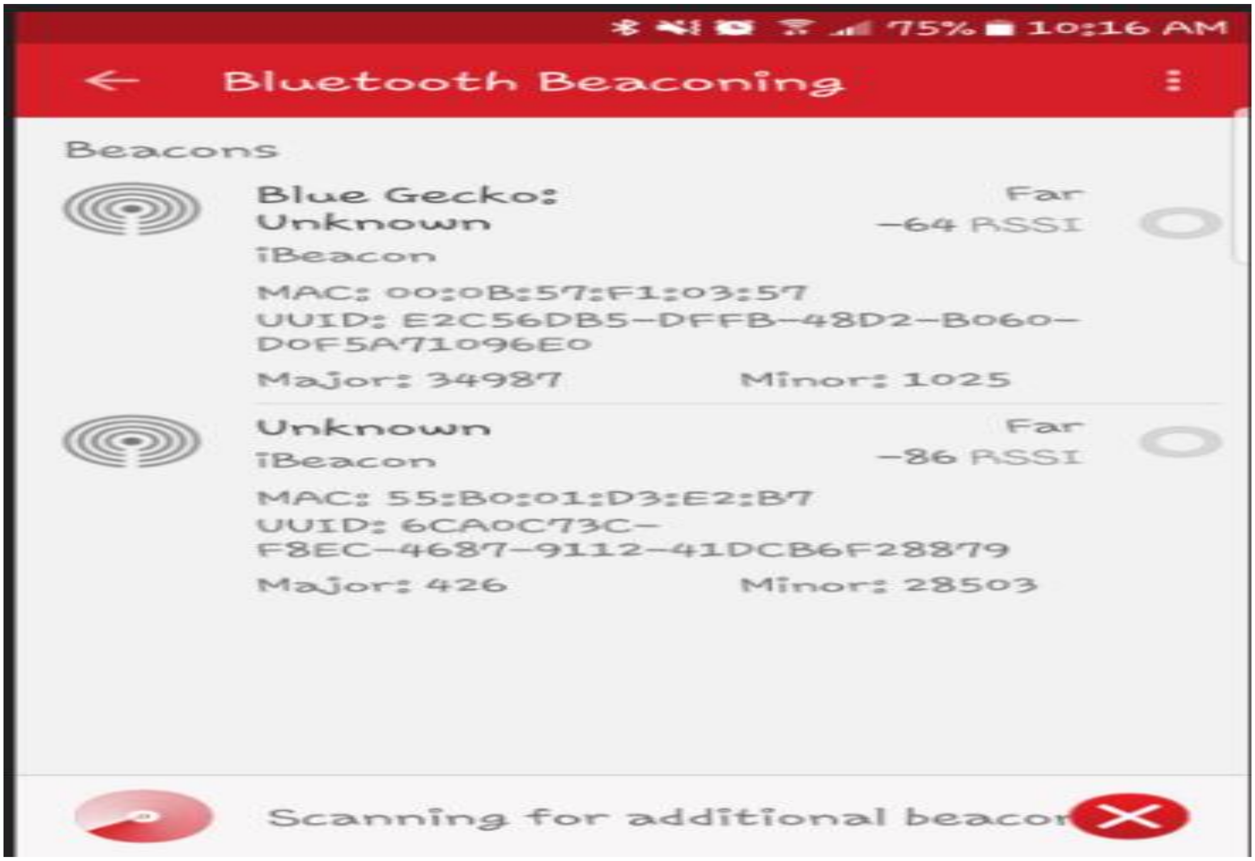
TESTING OF MQ2 SENSOR PERFORMED: -



Testing of the methane sensor MQ2 (Checked if it can detect alcohol in a sanitizer) by constructing this circuit on a general-purpose PCB board



The alarm circuit is built around NE555. If there is no alcohol in breath, the transistor will not conduct and hence NE555 will be reset otherwise it will be set and we get the sound from the speaker.



SAMPLE EXAMPLE TO TEST THE RANGE OF BLUETOOTH

BLUETOOTH BEACONS CAN BE SENT ON THE PHONE APP

WE OBTAINED A GOOD RANGE OF 6-7 meters.

SCOPE SHOT OF SIGNAL AT WIFI MODULE:

This is the noise measured at the WIFI module.

The frequency of its operation is 30.87Hz

LESSONS LEARNT:

- Learned to design our own 4- layer board
- We designed our own schematic by looking at the datasheet of each component used
- Understood the importance of having return path as close to the signal path as possible and hence designed our board as: Layer1: Power/Sig Layer2: GND PLANE 3: GND PLANE 4: Power/Sig
- Total current required by our circuit is obtained by adding up the current consumptions of each components and it is found to be 0.5Amps
- Learnt about a few more shortcuts in Altium:
  1) Pressing Shift key on a track part can be used to trace the entire routing path that is connected to it.
  2) Typing “JC” on the schematic or the Routing designs will open a small search window which can be used to search for a specific component on the schematic or routing design
- The characteristic impedance is 10 ohms
- No Blind vias should be used unless and until through hole vias can’t be placed as the advanced circuits considered those as pads and we had to connect wires externally to make the connections proper
- The pads of the IC BGM111 were underneath so it was difficult to solder, IC EFR32BG1 could have been used instead as it is easier to place
- While soldering the IC BME 280, due to reflow, the case of the air quality sensor was burnt so we replaced it with a new one and later got the correct ppm values of CO_2
- The LEDs were placed in the power path and hence there was a large voltage drop across them due to which the BGM111 was not receiving the proper voltage that it requires for operation (3.3v) so we had to take off those 2 LEDs. Our 3rd indicator LED was in the proper path and it glowed when the circuit was powered using USB.
- Initially we had problems, debugging our circuit and our board was not getting detected on the simplicity studios IDE but later when we connected the blind vias (GND) that turned out to be pads, with wires, our board got detected on the simplicity studio IDE we could program it and send the data over Bluetooth
- The switch that we had connected in our circuit, was default on and was turning off when pressed, we checked that with the help of connectivity test and replaced it with a new switch which was default off and thus our circuit worked
- We should have used surface mount MLCC caps instead of the through hole caps for decoupling to reduce the inductance in the path
- Our Bluetooth module worked but the WIFI module had a GND plane below it so there was a reflection coefficient of -1 which led to the cancellation of the WIFI signal.

- To place more Resistors that can be used to measure the current at various points in our board
- We successfully got the values from the dust sensor and air quality sensor (CO_2 in ppm) which closely matched the expected values given in the datasheet.
- Also, we could test the performance of methane sensor successfully by connecting it to a timer circuit and detecting the alcohol by placing a bit of sanitizer on the hand and the buzzer connected to the timer circuit rang indicating the detection of methane.

SOFTWARE CODE:

1) DUST SENSOR (USING ADC): -

/*Author: Sahana Sadagopan*/

```
#include "em_device.h"
#include "em_chip.h"

#include "hal-config.h"
#include "em_letimer.h"
#include "em_timer.h"
#include "em_cmu.h"
#include "em_gpio.h"
#include "em_emu.h"
#include "em_core.h"
#include "em_acmp.h"
#include "em_adc.h"

#include <stdint.h>
#include <stdbool.h>

void SleepMode(void){
	EMU_DCDCInit_TypeDef dcdcInit = EMU_DCDCINIT_DEFAULT;
	EMU_DCDCInit(&dcdcInit);
	EMU_EnterEM2(true);
}

void sleep(void)
{
	if (sleep_block_counter [0] > 0) {
	return;
	// Blocked everything below EM0, so just return
	}
	else if (sleep_block_counter [1] > 0) {
	EMU_EnterEM1();                                                   //
Blocked everything below EM1, enter EM1
	}
	else if (sleep_block_counter[2] > 0) {
	EMU_EnterEM2(true);
	// Blocked everything below EM2, enter EM2
	}
	else if (sleep_block_counter[3] > 0) {
	EMU_EnterEM3(true);                                               //
Blocked everything below EM3, enter EM3
	}
	return;
}
void blockSleepMode(uint32_t minimumMode)
{
	//instead of INT_Disable;
	CORE_DECLARE_IRQ_STATE;
	CORE_ENTER_ATOMIC();
```

```
	sleep_block_counter[minimumMode]++;
	CORE_EXIT_ATOMIC();
	//instead of INT_Enable;
}
void unblockSleepMode(uint32_t minimumMode)

{
	//instead of INT_Disable();
	CORE_DECLARE_IRQ_STATE;
	CORE_ENTER_ATOMIC();
	if(sleep_block_counter[minimumMode]>0){
		sleep_block_counter[minimumMode]--;
	}
	CORE_EXIT_ATOMIC();
	//instead of INT_Enable();
}


void ADC0_IRQHandler(void)
{
	uint32_t Flag = ADC_IntGet(ADC0);
	CORE_ATOMIC_IRQ_DISABLE();
	/*Clear the interrupting flags*/
	ADC_IntClear(ADC0, Flag);
	CORE_ATOMIC_IRQ_ENABLE();
}

void adc_Setup(void){
	//SleepMode();
	NVIC_DisableIRQ(ADC0_IRQn);
	uint8_t timeBaseValue =
ADC_TimebaseCalc(CMU_ClockFreqGet(cmuClock_HFPER));
	ADC_Init_TypeDef initadc={
			.ovsRateSel				= adcOvsRateSel2,
	//ADC_OvsRateSel_TypeDef
			.warmUpMode				=
adcWarmupKeepADCWarm,		//ADC_Warmup_TypeDef
			.em2ClockConfig			= 0,
			.timebase				= timeBaseValue,
	//uint8_t
			.prescale				= 54,
		//calculated for
			.tailgate				= 0
	};
	ADC_InitScan_TypeDef scaninit = ADC_INITSCAN_DEFAULT;
	ADC_Init(ADC0,&initadc);
	ADC_InitScan(ADC0,&scaninit);
	//ADC0->CMD= ADC_CMD_SCANSTART;
	ADC0->SCANCTRL |= ADC_SCANCTRL_CMPEN || ADC_SCANCTRL_AT_DEFAULT ||
ADC_SCANCTRL_REF_DEFAULT;
	ADC0->SCANCTRLX |= _ADC_SCANCTRLX_VREFSEL_VDDXWATT ||
ADC_SCANCTRLX_VREFATTFIX;
	ADC0->SCANMASK |=
_ADC_SCANMASK_SCANINPUTEN_INPUT0|_ADC_SCANMASK_SCANINPUTEN_INPUT2;
	ADC0->SCANINPUTSEL =
_ADC_SCANINPUTSEL_INPUT0TO7SEL_APORT1CH0TO7|_ADC_SCANINPUTSEL_INPUT0TO7SEL_AP
ORT2CH0TO7;
```

```
	ADC_IntEnable(ADC0, ADC_IEN_SCANCMP );
	NVIC_ClearPendingIRQ(ADC0_IRQn);
	NVIC_EnableIRQ(ADC0_IRQn);
	ADC_Start(ADC0, adcStartScan);

}
void adc_read(){
	uint32_t ADCdata;
	ADC0->SCANFIFOCLEAR |=ADC_SCANFIFOCLEAR_SCANFIFOCLEAR;
	ADCdata=ADC_DataScanGet(ADC0);
}
int main(void)
{
  /* Chip errata */
  CHIP_Init();
  /*To set in EM2*/
    //Low Noise initialization

    /*Start ADC setup*/
  adc_Setup();
  adc_read();

}
```

2) AIR QUALITY SENSOR CODE (USING I2C PROTOCOL): -

```
 /*
 * airquality.c
 *
 *  Created on: Apr 18, 2018
 *      Author: Sanika
 */

#include <airquality.h>

void i2c_slave_clear(void)
{
	for (int i = 0; i < 9; i++)
	{
		GPIO_PinOutClear(I2C0_SCL_PORT, I2C0_SCL_PIN);
		GPIO_PinOutSet(I2C0_SCL_PORT, I2C0_SCL_PIN);
	}
	return;
}

void i2c_bus_reset(void)
{
	if (I2C0->STATE & I2C_STATE_BUSY)
	{
		I2C0->CMD = I2C_CMD_ABORT;
	}
	return;
```

```
}

void i2c_init(void)
{
	cmu_i2c_init(cmuClkDiv_1);
	I2C0->ROUTEPEN = (I2C_ROUTEPEN_SDAPEN | I2C_ROUTEPEN_SCLPEN);
	I2C0->ROUTELOC0 = I2C_ROUTELOC0_SCLLOC_LOC14 |
I2C_ROUTELOC0_SDALOC_LOC16;
	I2C_Init_TypeDef i2c_init_parameter = I2C_INIT_DEFAULT;
	I2C_Init(I2C0,&i2c_init_parameter);
	i2c_bus_reset();
	GPIO_PinModeSet(I2C0_SCL_PORT, I2C0_SCL_PIN, gpioModeWiredAnd,
1);
	GPIO_PinModeSet(I2C0_SDA_PORT, I2C0_SDA_PIN, gpioModeWiredAnd,
1);
	i2c_bus_reset();
	i2c_slave_clear();
	I2C_IntClear(I2C0, (I2C_IFC_START | I2C_IFC_RSTART | I2C_IFC_ADDR
| I2C_IFC_TXC | I2C_IFC_ACK | I2C_IFC_NACK | I2C_IFC_MSTOP |
I2C_IFC_ARBLOST | I2C_IFC_BUSERR | I2C_IFC_BUSHOLD |I2C_IFC_TXOF |
I2C_IFC_RXUF | I2C_IFC_BITO | I2C_IFC_CLTO | I2C_IFC_SSTOP |
I2C_IFC_RXFULL | I2C_IFC_CLERR));
	I2C_IntDisable(I2C0, (I2C_IEN_START | I2C_IEN_RSTART |
I2C_IEN_ADDR | I2C_IEN_TXC | I2C_IEN_ACK | I2C_IEN_NACK |
I2C_IEN_MSTOP | I2C_IEN_ARBLOST | I2C_IEN_BUSERR | I2C_IEN_BUSHOLD
|I2C_IEN_TXOF | I2C_IEN_RXUF | I2C_IEN_BITO | I2C_IEN_CLTO |
I2C_IEN_SSTOP | I2C_IEN_RXFULL | I2C_IEN_CLERR));
	I2C0->IFC = 0x7FF;
	I2C0->IEN |= (I2C_IEN_ACK | I2C_IEN_RXDATAV);
	I2C_Enable(I2C0,true);
	return;
}

void i2c_stop(void)
{
	I2C0->CMD = I2C_CMD_STOP;
	return;
}

void i2c_ack(void)
{
	I2C0->IFC = I2C_IFC_ACK;
	return;
}

void i2c_nack(void)
{
	I2C0->CMD = I2C_CMD_NACK;
	return;
}

void i2c_start(void)
```

```
{
      I2C0->CMD = I2C_CMD_START;
      return;
}

void i2c_write_byte(uint8_t data_byte)
{
      I2C0->TXDATA = data_byte;
      while((I2C0->IF & I2C_IF_ACK) == 0);
}

uint8_t i2c_read_byte()
{
      while((I2C0->IF & I2C_IF_RXDATAV ) == 0);
      return I2C0->RXDATA;
}

void air_quality_init(void)
{
      uint8_t address=0;
      i2c_init();
      address  |= device_address_mask;
      address = address << 1;
      address &= write_mask;
      i2c_start();
      i2c_write_byte(address);
      //i2c_write_byte(power_on_command);
      i2c_ack();
      i2c_stop();
      return;
}

void air_quality_write(void)
{
      uint8_t address=0;
            i2c_init();
            address  |= 0x01;
            address = address << 1;
            address &= write_mask;
            i2c_ack();
            i2c_write_byte(data_1);
            i2c_ack();
            i2c_stop();
}
uint16_t read_channel(void)
{
      uint8_t data_byte= 0,data_byte_1=0,
data_byte_2=0,address=0,address_read=0,address_write=0;
      address |= device_address_mask_2;
      address_read = address << 1;
      address_write = address << 1;
      address_write &= write_mask;
```

```
	address_read |= read_mask;
	air_quality_init();
	//i2c_write_byte(address_write);
	//i2c_ack();
	i2c_write_byte(0x03);
	i2c_ack();
	i2c_stop();
	i2c_start();
	i2c_write_byte(address_write);
	i2c_ack();
	data_byte_1=i2c_read_byte();
	i2c_ack();
	data_byte_2=i2c_read_byte();
	data_byte = data_byte_1 <<8 | data_byte_2;
	i2c_nack();
	i2c_stop();
	return data_byte;
}

uint16_t airquality_read(void)
/*Function to read the air quality register*/

{
	uint8_t data_byte= 0,data_byte_1=0,
data_byte_2=0,address=0,address_read=0,address_write=0;
		address |= device_address_mask_2;
		address_read = address << 1;
		address_write = address << 1;
		address_write &= write_mask;
		address_read |= read_mask;
		air_quality_init();
		//i2c_write_byte(address_write);
		//i2c_ack();
		i2c_write_byte(0x02);
		i2c_ack();
		i2c_stop();
		i2c_start();
		i2c_write_byte(address_write);
		i2c_ack();
		data_byte_1=i2c_read_byte();
		i2c_ack();
		data_byte_2=i2c_read_byte();
		data_byte = data_byte_1 <<8 | data_byte_2;
		i2c_nack();
		i2c_stop();
		return data_byte;
/*The data_byte variable indicates the CO2 value in ppm*/
}
```

FUTURE SCOPE

The system can be further improved in the future to send alerts and notifications on the phone using a GSM module

CONCLUSION

We designed a prototype of weather station and got the readings of different sensors (real time data) and sent them over the silicon labs Bluetooth app using the BLE module in the circuit.

REFRENCES

- Air-quality sensor (https://www.digikey.com/product-detail/en/ams/CCS811BJOPD500/CCS811B-JOPD500TR-ND/6569311) (Temperature, humidity, CO2)
- 2) LPG, i-butane, propane, methane, alcohol, Hydrogen, smoke(https://www.digikey.com/product-detail/en/parallax-inc/605-00008/605-00008-ND/2666950)
- Dust sensor (https://www.mouser.com/ProductDetail/SharpMicroelectronics/GP2Y1010AU0F?qs=sGAEpiMZZMtWSrBd5SaE4KhLFfLe0cf3xOiZ9f5SMTw %3d)
- Pressure sensor (https://www.digikey.com/product-detail/en/bosch-sensortec/BME280/828-1063-1-ND/6136314)
- BGM-121 (Blue-gecko)
- Wi-Fi Module(http://electronut.in/an-iot-project-with-esp8266/)
- https://www.digikey.com/product-detail/en/silicon-labs/BGM121A256V1R/336-3810-1-ND
- https://people.ece.cornell.edu/land/courses/eceprojectsland/STUDENTPROJ/2009to2010/aps 243/asmyth_MEngCompilation/asmyth_MEng_Report.pdf