Hochschule Wismar University of
Applied Science Technology, Business and Design
Faculty of Engineering Department of
Electrical Engineering and Computer Science

PROJECT SEMINAR

:: Topic ::

Development of a power-measurement board in KiCad and programming ESP32-S3 microcontroller with Bluetooth Low Energy for data transmission to a mobile device

Mr.Divyesh Rajpara
(Matriculation Number : 419367)

Guided By

Prof. Dr.-Ing. Jens Kraitl
M. Eng. Andreas Hein

CONTENT

1. INTRODUCTION

In Project Seminar subject, student get opportunity to work on PCB designing KiCad designing software and also work on ESP32-S3 microcontroller. This year, main goal of this subject is to make design of Power-Measurement board in KiCad, which measure the voltage and current that generated from Photovoltaic panel. Measured device send measured current and voltage data to ESP32-S3 microcontroller board through SPI communication. Microcontroller share the data to mobile phone via Bluetooth connection. So that one can monitor generated updated data of current and voltage. The below figure represent basic block diagram of whole system.

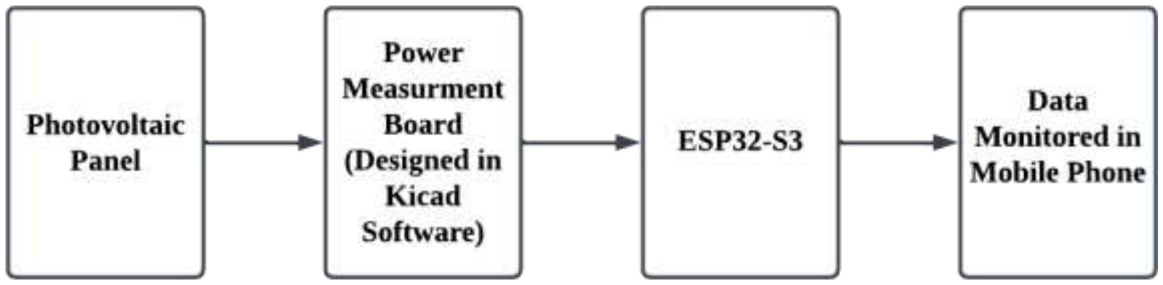
Photovoltaic Panel
Power Measurment Board (Designed in Kicad Software)
ESP32-S3
Data Monitored in Mobile Phone

Figure 1:- Block Diagram of System

2. LIST OF COMPONENT

1. INA 239:-

- The INA239 is an ultra-precise digital power monitor with a 16-bit delta-sigma ADC specifically designed for current-sensing applications. It measures shunt voltage, bus voltage and internal temperature while calculating current, power necessary for accurate decision making in precisely controlled systems. The main function of this device is to measure precise current and voltage, and it converts analog input to digital output [5].

2. ADuM120N and ADuM121N:-

- The ADuM120N and ADuM121N1 are dual-channel digital isolators. Combining high speed, complementary metal-oxide semiconductor (CMOS) and monolithic air core transformer technology, these isolation components provide outstanding performance characteristics superior to alternatives such as optocoupler devices and other integrated couplers. The maximum propagation delay is 13 ns with a pulse width distortion of less than 3 ns at 5 V operation. Channel matching is tight at 3.0 ns maximum. This device provides electrical isolation between various board parts, preventing voltage spikes, transients, or other disturbances from spreading over the isolation barrier. Digital isolator is able to maintain signal integrity by creating galvanic isolation between the measurement circuit and other component of system. Digital isolators offer interfaces between several voltage domains or communication methods. They may convert and isolate signals between various logic levels, assuring compatibility between the power measuring board and microcontrollers used in the system [6][7].
- The ADuM120N is a dual-channel digital isolator. It typically provides two distinct isolation channels, allowing for the isolation of two separate signals or data pathways.
- The ADuM121N is a quad-channel digital isolator. Up to four different signals or data pathways can be isolated using its four independent isolation channels.

3. ESP32-S3:-

- ESP32-S3 is a low-power MCU-based system on a chip (SoC) with integrated 2.4 GHz Wi-Fi and Bluetooth Low Energy (Bluetooth LE). It consists of high-performance dual-core microprocessor (Xtensa 32-bit LX7), a low power coprocessor, a Wi-Fi baseband, a Bluetooth LE baseband, RF module, and numerous peripherals. It has 45 programmable GPIOs. ESP32-S3 supports larger, high-speed octal SPI flash, and PSRAM with configurable data and instruction cache [8].

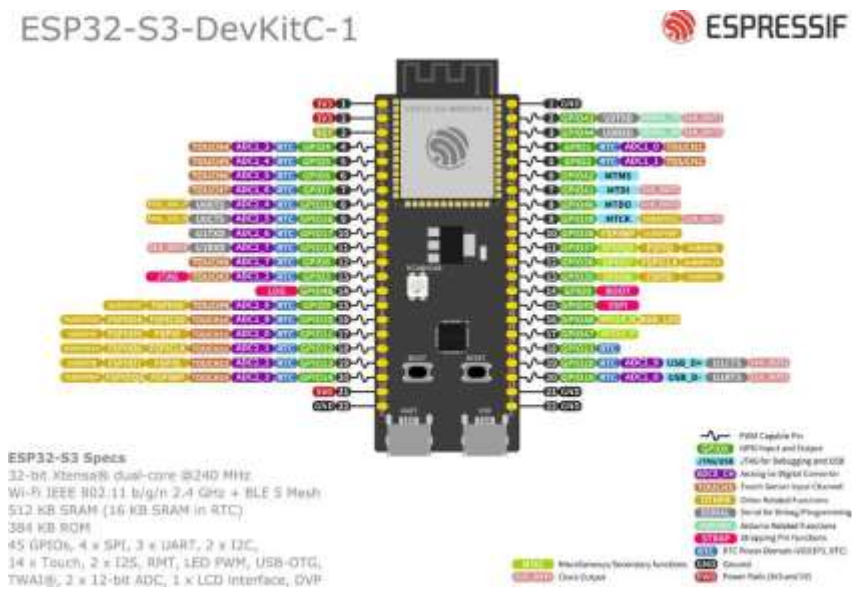
ESP32-S3-DevKitC-1
ESPRESSIF
ESP32-S3 Specs
32-bit Xtensa® dual-core @240 MHz
Wi-Fi IEEE 802.11 b/g/n 2.4 GHz + BLE 5 Mesh
512 KB SRAM (16 KB SRAM in RTC)
384 KB ROM
45 GPIOs, 4 x SPI, 3 x UART, 2 x I2C,
14 x Touch, 2 x I2S, RMT, LED PWM, USB-OTG,
TWAI®, 2 x 12-bit ADC, 1 x LCD interface, DVP

Figure 2:- Pin layout of ESP32-S3

3. KiCad

1. Brief Introduction of kiCad Software.

- KiCad is a free and open-source electronics design automation (EDA) suite. It features schematic capture, integrated circuit simulation, printed circuit board (PCB) layout, 3D rendering, and plotting/data export to numerous formats. KiCad also includes a high-quality component library featuring thousands of symbols, footprints, and 3D models [1][2].
- One can easily draw a circuit diagram in order to make Board layout in KiCad from pen paper based circuit, but person must know basic information and follow several rules for designing circuit such as,
  - **a. Component Selection:-** Selecting parts that are simple to find, satisfy your design specifications, and have corresponding footprints in KiCad's component libraries. Make sure the components you choose are appropriate for your project and are simple to get for manufacture. To match components that are not in the library, you may need to build unique footprints in KiCad.
  - **b. Component Placement:-** Place components on the PCB layout systematically to improve various kinds of factors like signal integrity, heat management, and manufacturing simplicity. Look of things like route simplicity, soldering accessibility, and component orientation. Component placement should be well-planned to minimize signal interference, ensure effective heat dissipation, and make assembly and testing easier.
  - **c. Design Rules:-** Create and set up KiCad design rules that are unique to your project. These regulations specify criteria like clearances, minimum trace widths, and other limitations on manufacture. These design guidelines must be followed in order to avoid problems during PCB production and assembly. You may make sure that your PCB layout complies with industry standards and manufacturing capabilities by defining appropriate design guidelines.

- Basic Terminology of KiCad:-

I. Schematic Editor:- A schematic designed with KiCad is more than a simple graphic representation of an electronic device [1]. A schematic mainly consists of symbols, wires, labels, junctions, buses and power symbols. When designing schematic a person must follow grid because wires connect with other wires or pins only if their ends coincide exactly. If a person not follow grid rule, then their schematic design have improper connection between wires and component. Therefore it is very important to keep symbol pins and wires aligned to the grid. The default grid size is 50 mil or 1.27 millimeters in KiCad. Symbols are added to the schematic from symbol libraries, after that all symbols connected with each other through wires and symbols also connected to necessary power symbols such as Ground and Power flag. After the schematic is made, the Electrical Rules Check (ERC) tool checks for certain errors in your schematic, such as unconnected pins, unconnected hierarchical symbols, shorted outputs or other illegal connections, etc. After rectifying errors a one is able to set of connections and footprints is imported into the PCB editor for designing a board [3].

- ❖ Explanation of Schematic design:- The following design is schematic design of Power measurement board. This design contains various components such as INA239, ADuM120N, ADuM121N, Male and Female connectors, Solder nail, 3-Pole screw connection terminals, different value Resistors and Capacitors. Photo-Voltaic panels, ESP32-S3 microcontroller, Synchronous buck converter and Synchronous step-down controller are externally connected to the Power measurement board through connection terminals. Generated voltage and current passed to the INA239 through R-C network. INA239 is generally used to sense voltage and current. Here, R1 resistor is parallel connected to 2 R-C circuit. This R-C circuit act as low-pass filter, which attenuating high frequency.

  Here, Calculation of Shunt resistor,

  $R_{shunt} = V_{max}/I_{max}$

  $V_{max} = \pm$ 163.4 mv (from the datasheet)
  $I_{max} = I_{oc} = 3.4$ A (from the datasheet)

  $R_{shunt} = 163.4mV / (3.4\ A)$

  $R_{shunt} \sim= 0.4805\Omega$

  At input side of INA239, voltage and current measured. Internally, these measured analog value data converted to digital data. INA239 is communicated with microcontroller through SPI protocol. In INA239, the objective of connecting a resistor between the INA239's power supply pin (Pin 6) and alert pin (Pin 3) is to monitor the power supply's status and provide an alert signal. Commonly known as a pull-up resistor, this resistor. Pull-up resistors are used to ensure that the alert pin (Pin 3) stays at a known voltage level (often the power supply voltage) even when the alert output is not actively driving it down. Pull-up resistors are connected between the power supply pin (Pin 6) and the alert pin (Pin 3). The alert pin is kept from floating, which could result in false readings or undefinable states, thanks to this configuration, which also helps to maintain a stable reference voltage. The INA239 actively pulls the alert pin when an alert circumstance arises.

- ❖ INA239 serially sent data to microcontroller through digital isolators. Voltage measurement channels are frequently found on power measuring boards because of different voltage measurement channel, these different channels generate interference in nearby channels which lead unwanted interferences in power measurement board. To avoid unwanted interference, electrical isolation is required in power measurement board. So that, digital isolator used in power measurement circuit. Because of SPI communication between microcontroller and current sensing device 2 ADuM120N and 1 ADuM121N digital isolator used. ADuM120N and ADuM121N isolator isolate 2 parts of power detection board, but these isolators transmit data between INA239 and ESP32-S3 controller. These 2 digital isolator used Magnetic Coupling technique. Signal transmission between the input and output sides is accomplished by means of a transformer-like device. Signals are transferred through the isolation barrier using magnetic fields.

- Output signal is transferred to the ESP32-S3 through screw connection terminals and J1socket and J2 pin from digital isolators. Here, Two capacitors are placed next to the isolator, connected in parallel, and have significant value discrepancies. The capacitor acts as a reservoir, absorbing extra charges when the voltage goes above its rated amount. The capacitor releases the collected charges when the voltage drops in order to keep a steady supply. Low-frequency voltage variations are handled by a 10 uF capacitor, whereas high-frequency voltage changes are handled by a 100 nF capacitor. On the output side of the isolator, a pull-up resistor is frequently used in order to establish a trustworthy digital connection between the two sides. By ensuring that the output signal is pulled to a specific voltage level while it is in the high state or not being actively pushed, this resistor helps. When the output signal is in the high state or not being actively pushed, this resistor helps to ensure that it is pulled to a specific voltage level. When the signal is not being driven actively, the pull-up resistor aids in maintaining a constant voltage level on the output side. It ensures a known and dependable signal level by preventing output from drifting or being impacted by noise.

- In the design of a measurement board, two supply voltages and two ground polygons are commonly utilized. There are two separate power supply used to protect analog and digital circuits from noise and interference. To ensure precision in measurements free from any digital noise or voltage changes, analog components require a stable and dependable power source. For digital circuits, also steady power source required in order to operate digital components. Because of 2 different circuits, 2 different voltage sources required to operate both circuit in desire way in.

- Here, 2 different polygons also used to avoid crosstalk and interference between analog and digital components can be prevented by designing separate ground planes for each kind of component. It aids in preventing the development of ground loops, which can corrupt signals and produce obtrusive noise.

- Operating range of power supply of this board is 2.7V to 5.5V. With help of this Power Measurement Board one can measure -0.3 to 85V input voltage and up to 10A current

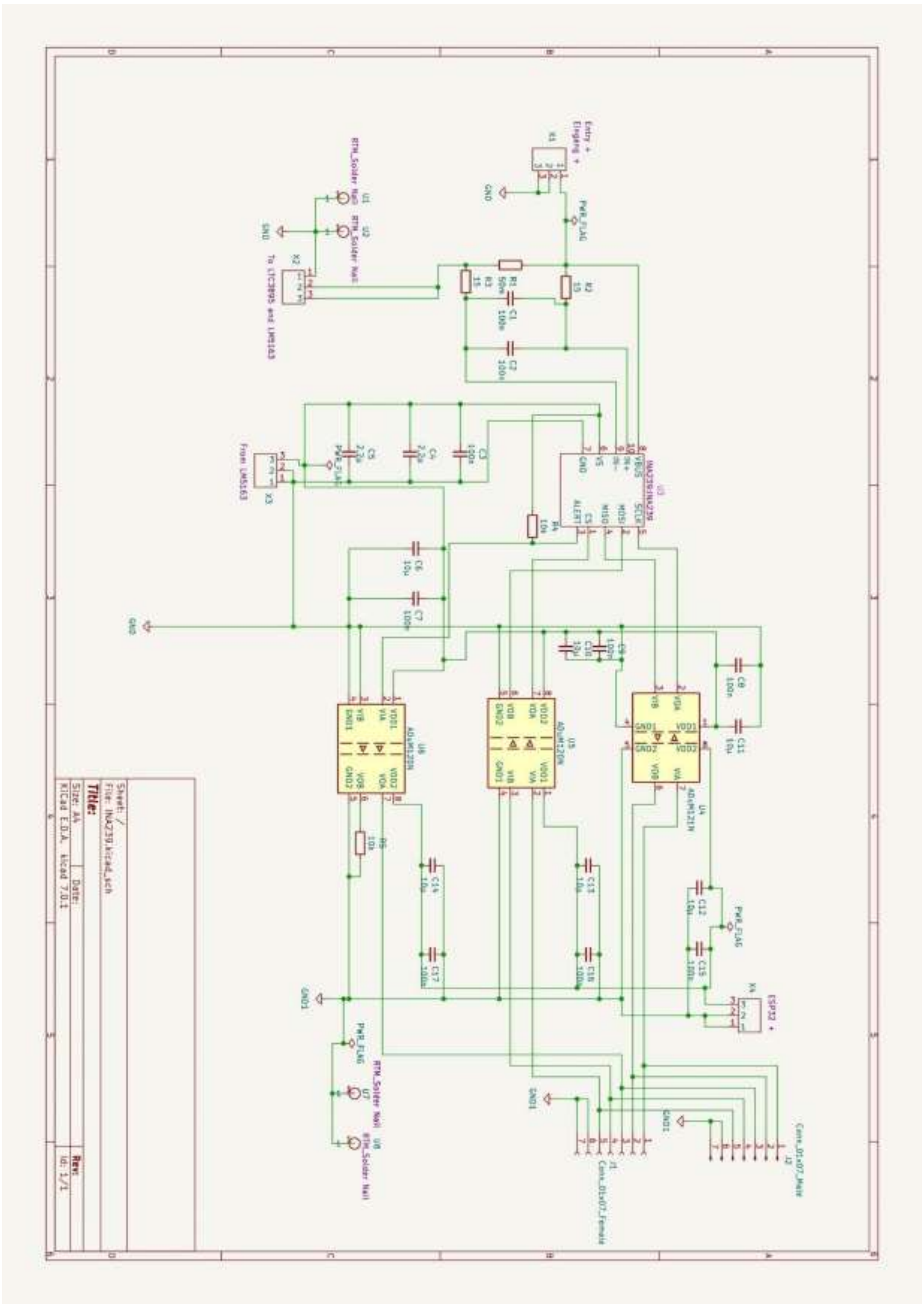
Figure 3:- Schematic design of Power Measurement Board

II. Symbol Editor :- One can create own symbol library, create a new symbol library and also edit existing symbols for their specific Project [1]. A symbol represent appearance of electrical component. In the symbol editor, there are various tools as one can place a pin according to their requirement of symbols, also assign electrical type, number, and pin type to pin and make the desired shape of the symbol. With the help these tools one can manage to design their symbols for their project. Here, several examples of created symbols in symbol editor.

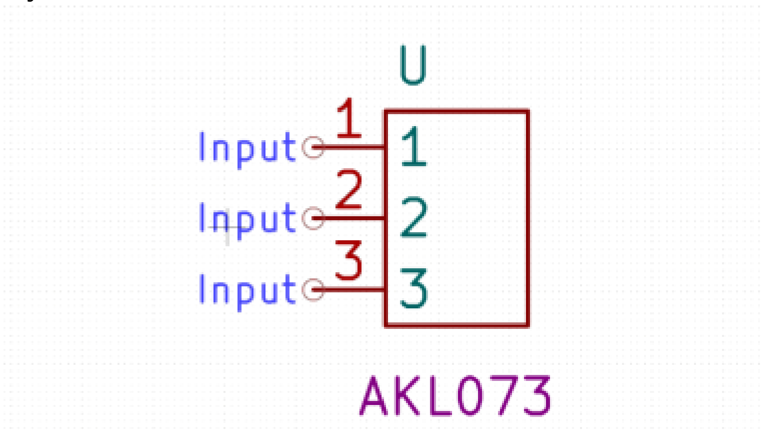

Figure 4:- Symbol design of AKL073

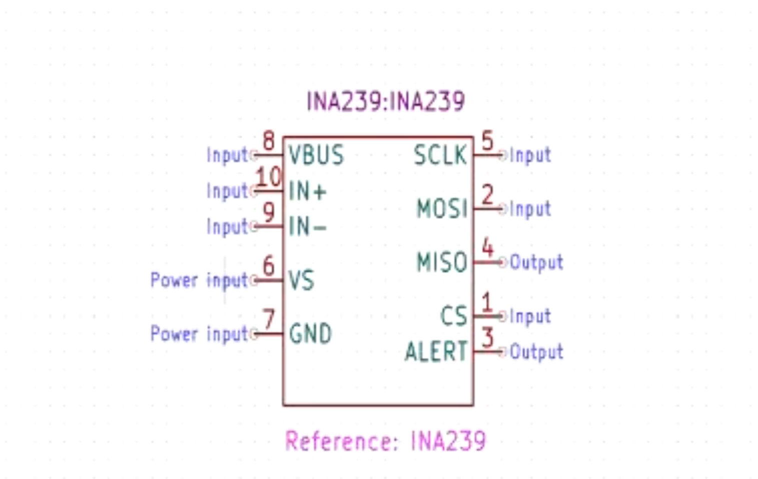

Figure 6:- Symbol design of INA239

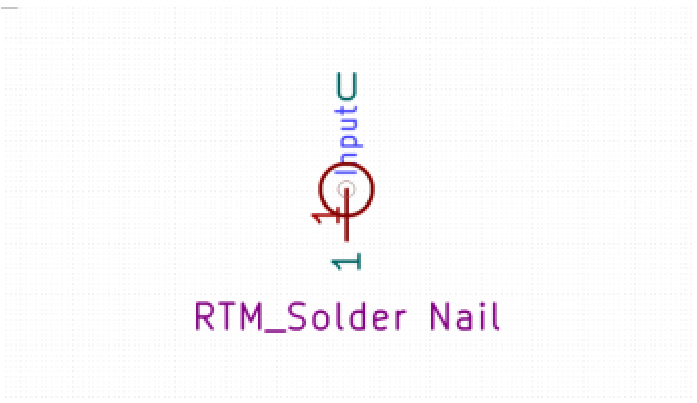

Figure 5:- Symbol design of RTM (Solder Nail)

III. Footprint Editor:- With help of this editor anyone is able to create a new footprint and edit existing footprint. Footprints often represent physical electrical components [1]. Footprint editor have some different tools. Using Footprint template tool, one has to choose suitable template for footprint based on your component's package type and one can manage to modify component's physical dimensions and pin arrangement as per component' datasheet. After creating footprint, person has to add Silkscreen markings on footprint to help in component placement and identification, then add Courtyard boarder which represents the boundary area around the component. After successfully creating footprint, one have to save to library and assign footprint library to specific name of symbol in symbol editor. Here the example of 2 different type of footprint such as SMD and Through-Hole.

- SMD:- In this type of footprint, one can be soldered component directly onto the surface of the PCB.

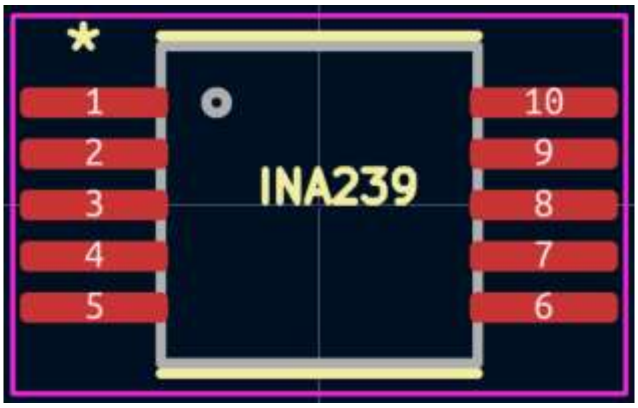

Figure 7:- Footprint design of INA239

- The rectangle uses the F.courtyard layer, and same the square uses the F.fab layer. Two lines are made up of the F.silkscreen layers, which is 3.5mm parallel to each other. Pads have round rectangle shape of 1.45 mm and 0.3mm with corner radius 0.075mm and 25% corner size.

- Through-Hole:- In this type of footprint, one have to pass component pins or leads through-hole and can be soldered on the back side of the PCB.

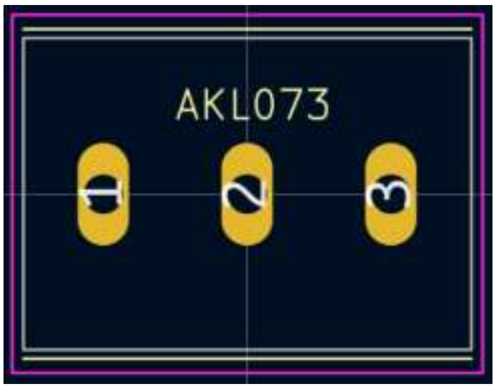

Figure 8:- Footprint design of AKL

- The rectangle uses the F.courtyard layer and smaller rectangle uses the F.fab layer. Two lines are made up of the F.silkscreen layers, which is 10.26mm parallel to each other. 3 Pads have oval rectangle shape of 1.8 mm and 3.4mm with diameter of 1.4mm.

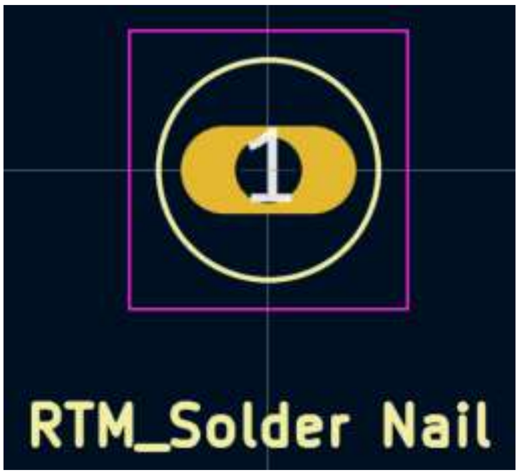

Figure 9:- Footprint design of RTM.

- The rectangle uses the F.courtyard layer. Circle with 2mm radius uses F.silkscreen layers with 0.1mm line width. Pads have oval rectangle shape of 1.6 mm and 3.2mm with diameter 1.21mm.

IV. PCB Editor:- With the help of this editor, anyone can create a new footprint and edit an existing footprint. Footprints often represent physical electrical components [4]. Footprint editors have some different tools. Using the Footprint template tool, one has to choose a suitable template for footprint based on your components:- In this editor, circuit designers create a Board layout or can modify the Board layout. If one has already created a schematic design of the circuit, then one can easily create a board layout from the schematic design. A person needs to import the netlist file into the PCB editor from Schematic. Because the netlist file contains information about the connection between components, which is important for Board layout. After importing the file, the designer should define the shape of the PCB by using a drawing tool. Then, one can place the component on the PCB layout. For component placement, designers have to consider some factors such as signal integrity, thermal management minimizing trace length, etc. In the next step, routing is started to connect components. In the routing task, the user should satisfy various rules, The posture of the track should not be 90 degrees, Width of the track depends on the connection between components. Track width is more in high power-consuming components compared to less power-consuming components. Some components can't connect on the top layer, because of more connection. But with the help of Vias, one can connect those components through different layers. After routing, one can create polygons on the PCB layout that will be filled with copper for power and ground connection. Polygons is providing a low-impedance path for power distribution and ground reference plane. Later, the designer should run Design Rule Check. DRC checks the connection between components, trace width, and other electrical rules. After successfully creating a Board layout, one can view their board in 3D view using the 3D viewer.

- Many calculations are made during the board design phase of a project to ensure the functionality, reliability, and legality of the design such as,

  - Trace Width and Current Capacity:- To avoid excessive resistance, voltage loss, and overheating, determine the appropriate trace width based on the maximum current.

  - Decoupling Capacitor selection:- Determine the necessary capacitance for decoupling capacitors based on the desired voltage ripple and current demand.

  - Thermal Analysis:- Using calculations for power dissipation, thermal resistance, and junction temperature, evaluate component temperature growth.

- Clearance and Creepage distances:- Determine the bare minimum separation distances necessary to provide electrical safety and insulation between high-voltage components and conductive traces.

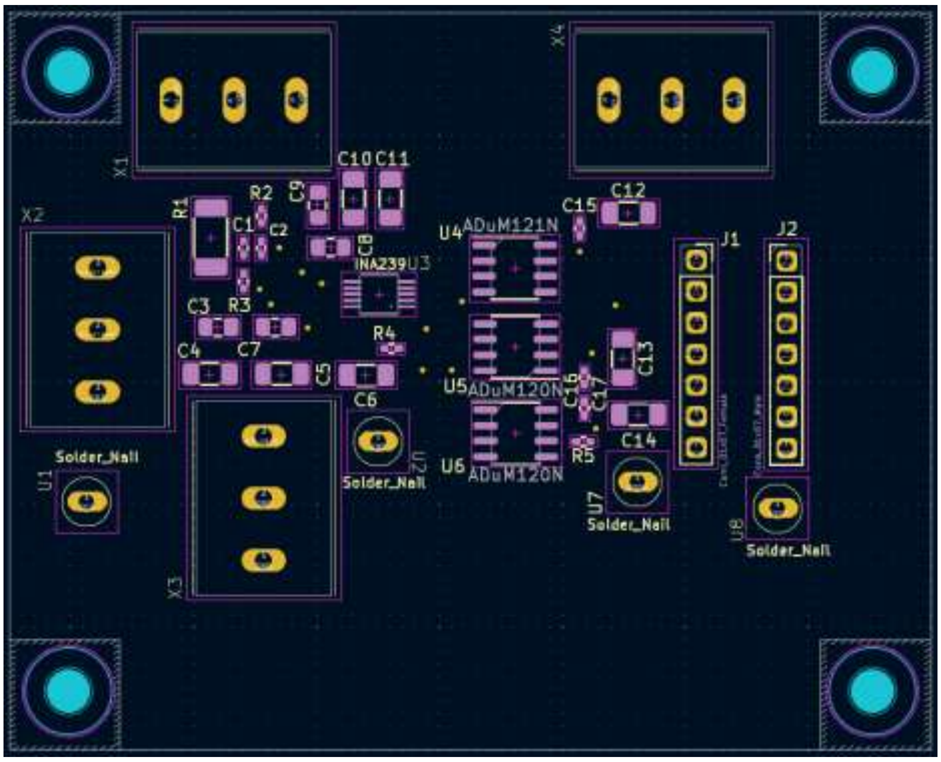

Figure 10: PCB Design without routing

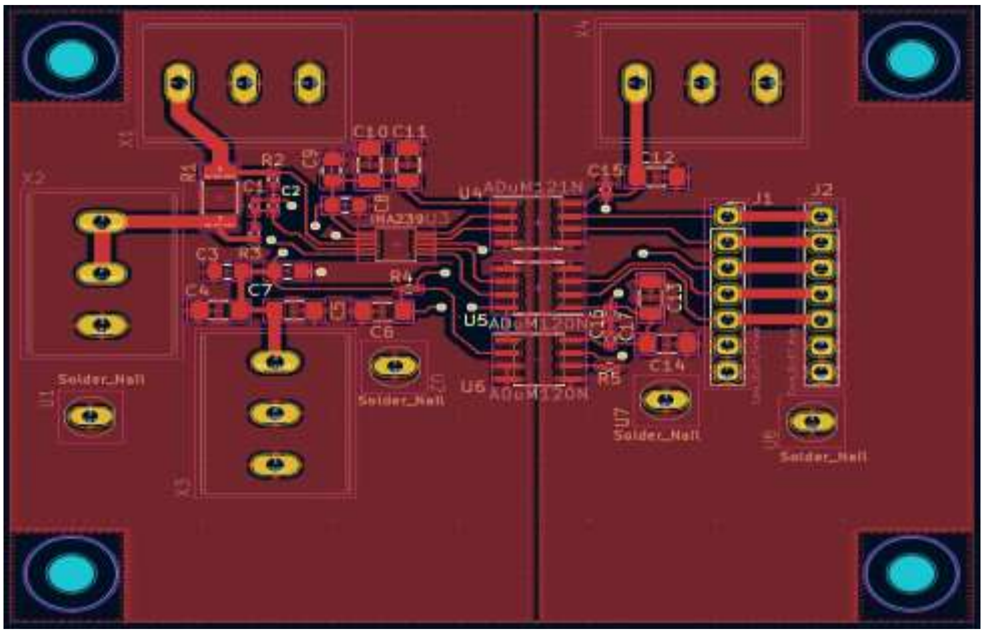

Figure 11: PCB Design with routing

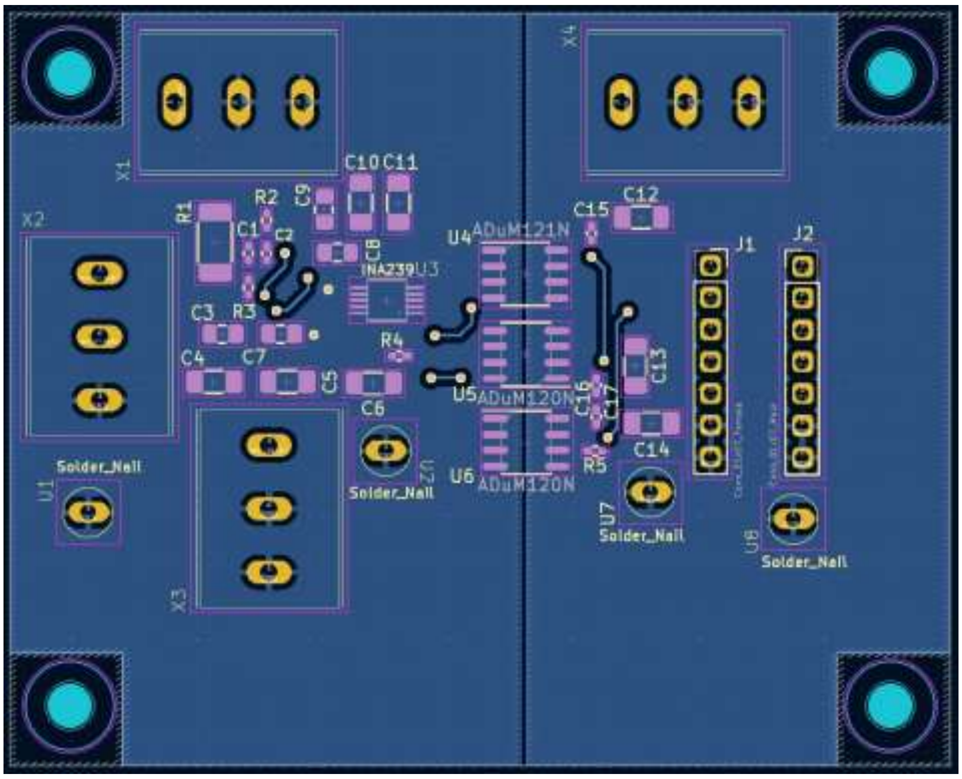

Figure 12:- Bottom layer design of PCB with B.cu

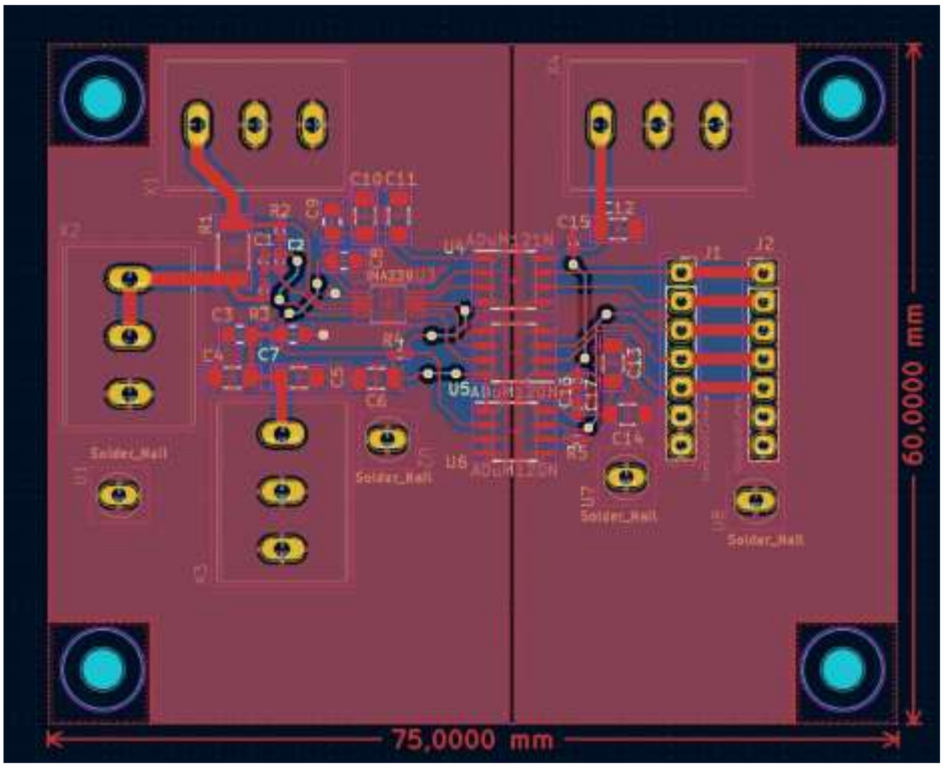

Figure 13:- PCB board layout Power Measurement Board

•

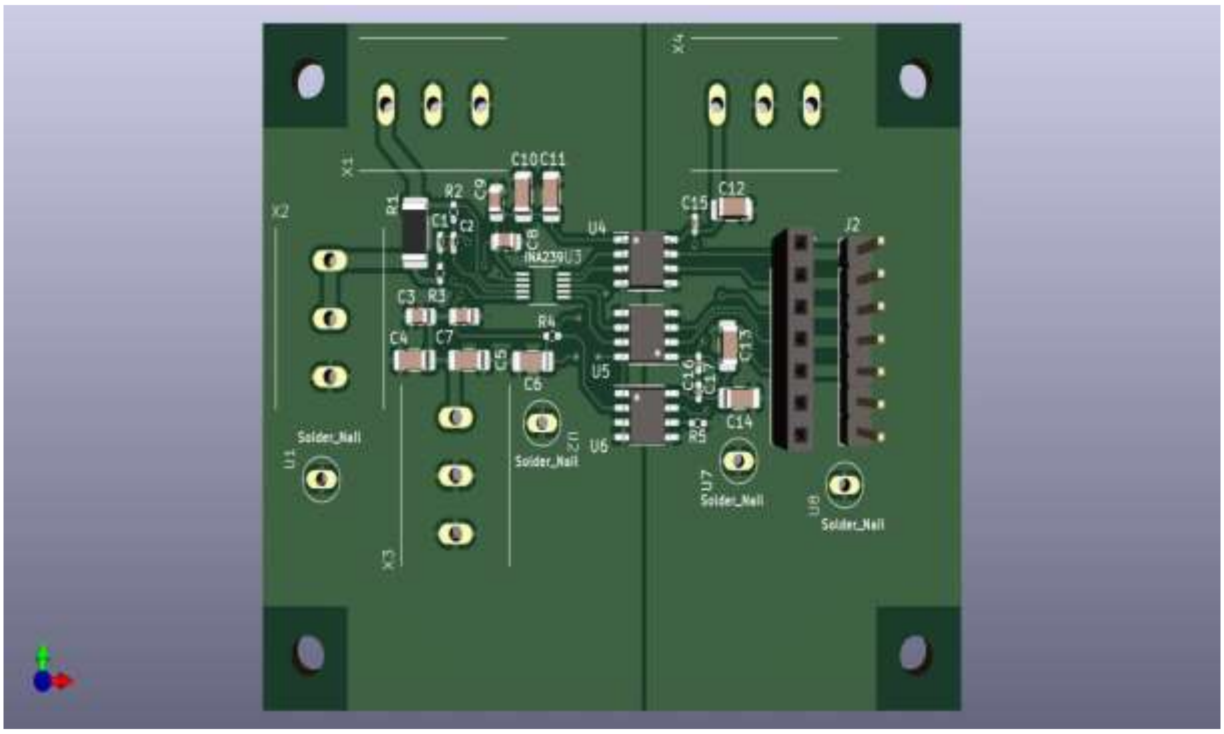

Figure 14:- Top view of 3D design of PCB

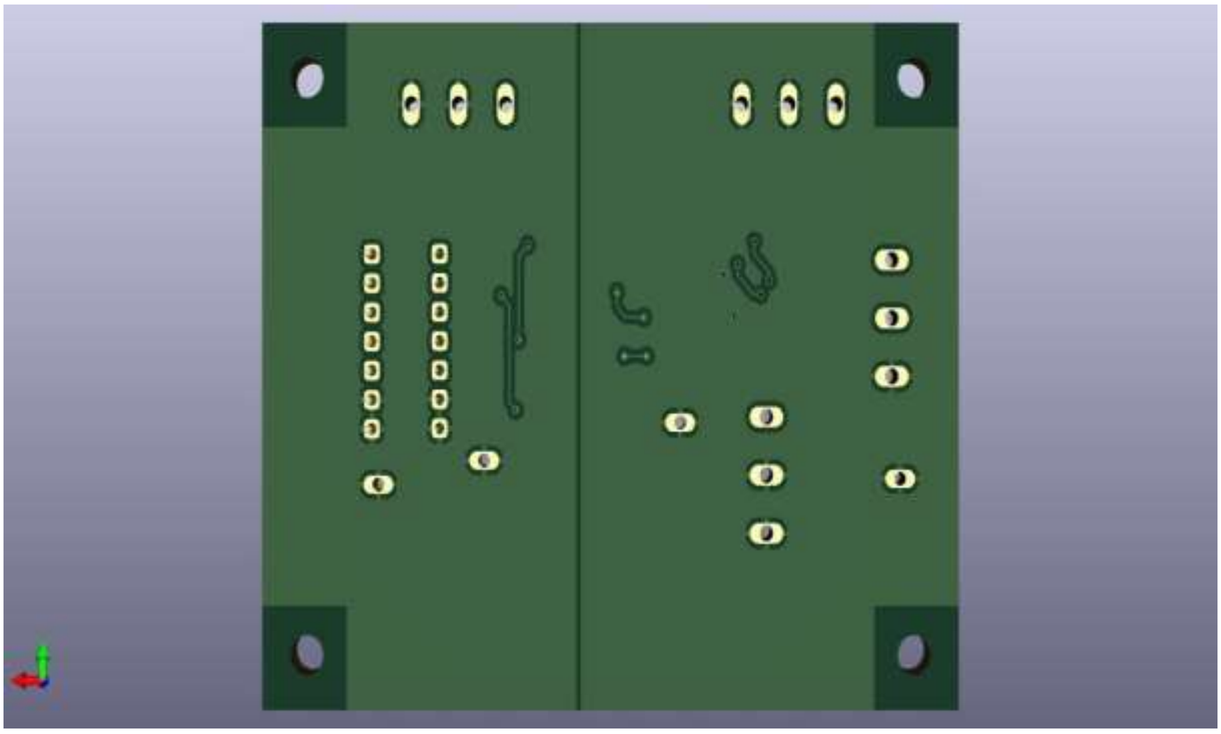

Figure 15:- Bottom View of 3D design of PCB

4. SERIAL PERIPHERAL INTERFACE

SPI is a synchronous serial communication interface used in electronic systems to connect peripheral devices, microcontrollers, and microprocessors. It enables master-slave architecture between devices, allowing data interchange. There are mainly three types of SPI of configurations are:

- ❖ Single slave configuration.
- ❖ Multi slave configuration.
- ❖ Daisy chain configuration.

1. Single slave configuration:- In this case, the master device is connected to a single slave. The master device manages communication with the slave device through the SPI bus. This is how SPI is set up in it is most basic and standard form.

2. Multi slave configuration:- In this system, a single master device connects with a number of slave devices. Each slave device has a distinct slave select (SS) line, which the master controls. The master identifies a certain slave by asserting the associated slave select line, and the selected slave and master can then converse.

3. Daisy chain configuration:- The MOSI line of the master (Master Output Slave Input) is connected to the MOSI input of the first slave. The MOSI output of the first slave is connected to the MOSI input of the second slave, and so on. The MISO (Master Input Slave Output) line is wired in the other direction and is similar to this. The master controls communication with each slave by selectively allowing or preventing data flow in the daisy chain.

5. PROGRAMMING

➢ With which resolution can PV module current, PV module voltage and PV module power be measured? Give the calculations for this in your report and explain and justify your calculations.

❖ VSHUNT and VBUS registers have 16-bit digital output whereas 12-bit digital output in the DIETEMP register. The maximum predicted current is used to determine the value of CURRENT_LSB. and it expressly specifies how the current register will be resolved. While the highest resolution is produced by the smallest CURRENT_LSB value.

Here, calculation for current.

$$\text{Resolution} = \frac{Maximum\ expected\ range}{Bit\ length}$$

$$\text{Current\_LSB} = \frac{10}{2^{15}}$$

$$= 305.176 \text{ mA / LSB}$$

This is for Power,

Power (w) = Current_LSB * 0.2

= 305.176 mA * 0.2

= 61.035756

- How BLE connection is established between ESP32-S3 and smartphone?

  - Establishing a BLE connection with a smartphone requires first initializing the ESP32 microcontroller's BLE capability.
  - Initialize BLE: Set up the ESP32 microcontroller's BLE capabilities by configuring it to behave as a BLE peripheral.
  - Define Services and Characteristics: Define the features and services that will be utilized to exchange data between the microcontroller and smartphone. Services are a group of similar features, whereas characteristics are the specific data items that are shared.
  - Advertise the BLE Peripheral: Make the microcontroller discoverable by the smartphone by broadcasting information about it, such as the device name and supported services. The smartphone can now recognize and connect to the microcontroller thanks to this.
  - Handle BLE Events: Put in place event handlers to react to different BLE events as they happen along the communication process. These events include requests for connections, breaks in connections, requests for data reads and writes, and alerts.
  - Establish a Connection: Accept the connection request from the smartphone and establish a BLE connection between the microcontroller and smartphone. This enables bidirectional communication between the two devices.
  - Data Exchange: To exchange data between the microcontroller and smartphone, make use of the provided services and characteristics. In order to do this, either data must be read from the smartphone or sent from the microcontroller to the smartphone.
  - Handle Disconnection: If necessary, prepare for a fresh connection and gracefully handle disconnection events by cleaning up your resources. In doing so, the microcontroller is guaranteed to be able to manage unforeseen disconnections and resume normal operation.
  - Implement Security Measures (Optional): You can decide to add more security measures for the BLE connection depending on the project's requirements. To safeguard the integrity and confidentiality of the sent data, this may involve data encryption, secure pairing, or access control techniques.

➢ **Programme:-**

```
#include <Arduino.h>
#include <SPI.h>
#include <pins_arduino.h>
#include <driver/spi_master.h>
#include <driver/spi_common.h>
#include <esp32-hal-spi.h>

#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
#include <Wire.h>

#define SERVICE_UUID "00000000-0000-0000-0000-000000000000"
#define CHARACTERISTIC_UUID_TX "00000000-0000-0000-0000-000000000001"
#define CHARACTERISTIC_UUID_RX "00000000-0000-0000-0000-000000000002"

#define DEBUG       1 // BLE
#define ADT_DEBUG   0 // Restart ESP32-S3 caused by ADT

// INA239
// Datamode von INA
#define DATAMODE_INA      1 // CPOL=0; CPHA=1 -> MODE
// config_Reg_INA
#define RST_CONFIG_REG     0x000000 // system restart
#define RD_CONFIG_REG      0x010000 // read CONFIG_REG (address=oh)
#define WR_CONFIG_REG      0x000040 // read CONFIG-REG: RST(15)=0, CONVDL(13-6)=1=(2ms),
ADCRANGE(4)=163.84mV
// ADC_Config_Reg_INA
#define WR_ADC_CONFIG_TRIG_REG 0x047B6C
//#define WR_ADC_CONFIG_CONT_REG 0x04FB6C // MODE = 0xF Continuous VCT, CNVT=1052us.
AVG=128
//#define WR_ADC_CONFIG_CONT_REG_50us_AVG1 0x04F000// MODE = 0xF Continuous VCT, CNVT=50us
AVG=1
//#define WR_ADC_CONFIG_CONT_REG_50us_AVG64 0x04F003
#define WR_ADC_CONFIG_CONT_REG_540us_AVG64 0x04F923
#define WR_ADC_CONFIG_TRIG_REG_540us_AVG64 0x047923
#define WR_ADC_CONFIG_TRIG_REG_540us_AVG1 0x047000
#define RD_ADC_CONFIG_REG  0x050000
// Shunt_Cal_Reg_INA
#define WR_SHUNT_CAL_REG 0x080EA6
#define RD_SHUNT_CAL_REG 0x090000
// VShunt_Reg_INA
#define RD_VSHUNT_REG 0x110000
// VBus_Reg_INA
#define RD_VBUS_REG  0x150000 // voltage+VBUS_REG(decimal)*3125uV/LSB
// MEasure_Reg_INA
```

```
BLEServer *pServer = NULL;
BLECharacteristic *pTxCharacteristic;
bool deviceConnected = false;
bool oldDeviceConnected = false;

const int INA_ADC_BUSY_pin = 2;  // Replace 2 with the actual pin number

void getValues(void);

void spiCommand24(SPIClass *spi, uint32_t data, uint8_t spi_datamode) {
  SPI.beginTransaction(SPISettings(1000000, MSBFIRST, DATAMODE_INA));
  digitalWrite(SPI.pinSS(), LOW);
  SPI.transferBits(data, NULL, 24);
  digitalWrite(SPI.pinSS(), HIGH);
  SPI.endTransaction();
}

void spiCommand32(SPIClass *spi, uint32_t data, uint8_t spi_datamode) {
  SPI.beginTransaction(SPISettings(1000000, MSBFIRST, DATAMODE_INA));
  digitalWrite(SPI.pinSS(), LOW);
  SPI.transfer32(data);
  digitalWrite(SPI.pinSS(), HIGH);
  SPI.endTransaction();
}

uint8_t ctrl_FLAG_REG = 0;  // Replace uint8_t with the appropriate data type

void IRAM_ATTR onINA_ready() {
  detachInterrupt(INA_ADC_BUSY_pin);
  ctrl_FLAG_REG |= 0x01;
}

class MyServerCallbacks : public BLEServerCallbacks {
  void onConnect(BLEServer* pServer) {
    deviceConnected = true;
  }

  void onDisconnect(BLEServer* pServer) {
    deviceConnected = false;
  }
};

class MyCallbacks : public BLECharacteristicCallbacks {
  void onWrite(BLECharacteristic *pCharacteristic) {
    std::string rxValue = pCharacteristic->getValue();
    if (rxValue.length() > 0) {
      Serial.println("*********");
      Serial.println("Received Value: ");
      for (int i = 0; i < rxValue.length(); i++) {
        Serial.print(rxValue[i]);
      }
      Serial.println();
      Serial.println("*********");
    }
  }
};

void setup() {
  // put your setup code here, to run once:
  BLEDevice::init("Project-Seminar-SS2023");
  pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());

  BLEService *pService = pServer->createService(BLEUUID(SERVICE_UUID));

  pTxCharacteristic = pService->createCharacteristic(
    BLEUUID(CHARACTERISTIC_UUID_TX),
    BLECharacteristic::PROPERTY_NOTIFY
  );

  pTxCharacteristic->addDescriptor(new  BLE2902());

  BLECharacteristic *pRxCharacteristic = pService->createCharacteristic(
    BLEUUID(CHARACTERISTIC_UUID_RX),
    BLECharacteristic::PROPERTY_WRITE
  );
  pRxCharacteristic->setCallbacks(new  MyCallbacks());

  pService->start();

  pServer->getAdvertising()->start();
  Serial.println("Waiting for a client connection to notify.");
}

// Define and initialize the SPI object
SPIClass mySPI2(HSPI);


void loop() {
  // put your main code here, to run repeatedly:
  if ((ctrl_FLAG_REG & 0x10) == 0x10) {
    spiCommand24(&mySPI2, WR_ADC_CONFIG_TRIG_REG_540us_AVG64, DATAMODE_INA);
    ctrl_FLAG_REG = (ctrl_FLAG_REG & (~0x10));
    delay(10);
  }

  const uint8_t RD_DIETEMP_REG = 0x10;  // Replace 0x10 with the correct value
  uint16_t receivedVal = 0;  // Replace uint16_t with the appropriate data type
  const uint8_t RD_POWER_REG = 0x20;  // Replace 0x20 with the correct value
  uint16_t tmpDIETEMP_INA_raw = 0;

  float rawDataBuff[5] = { 0.0, 0.0, 0.0, 0.0, 0.0 };
  if ((ctrl_FLAG_REG & 0x01) == 0x01) {
    spiCommand24(&mySPI2, RD_DIETEMP_REG, DATAMODE_INA);
    tmpDIETEMP_INA_raw = (receivedVal >> 15);
    rawDataBuff[0] = (tmpDIETEMP_INA_raw * 25);
    spiCommand24(&mySPI2, RD_VBUS_REG, 1);
    uint16_t tmpVBUS_INA_raw = receivedVal;
    rawDataBuff[1] = tmpVBUS_INA_raw * 30;
    spiCommand24(&mySPI2, RD_CONFIG_REG, DATAMODE_INA);
    uint16_t tmpCURR_INA_raw = receivedVal;
    rawDataBuff[2] = tmpCURR_INA_raw * 5;
    spiCommand24(&mySPI2, RD_POWER_REG, DATAMODE_INA);
    uint16_t tmpPWR_INA_raw = receivedVal;
    rawDataBuff[3] = tmpPWR_INA_raw * 23;
  }

  if (deviceConnected) {
    Serial.println("\nDevice connected..\n");
    delay(10);

    struct {
      char Data_Block_Caption[100];
      char PV_Module_Voltage[50];
      char PV_Module_Current[50];
      char PV_Module_Power[50];
      char Meas_Box_Temperature[50];
      char PV_Error_Message[50];
    } measurement_data;

    char dtostrf_buff[20];

    memcpy(measurement_data.Data_Block_Caption, "\n\n PV_Module Measurements: \n\n",
strlen("\n\n PV-Module Measurements: \n\n") + 1);
    memcpy(measurement_data.Meas_Box_Temperature, "\n Measured box-temperature:",
strlen("\n Measured box-temperature:") + 1);

    dtostrf(rawDataBuff[0], 7, 1, dtostrf_buff);
    strcat(measurement_data.Meas_Box_Temperature, dtostrf_buff);
    strcat(measurement_data.Meas_Box_Temperature, "°C");
    memset(dtostrf_buff, 0, sizeof(dtostrf_buff));

    memcpy(measurement_data.PV_Module_Voltage, "\nPV-Module output voltage:", strlen("\nPV-
Module output voltage:") + 1);
    dtostrf(rawDataBuff[1], 7, 2, dtostrf_buff);
    strcat(measurement_data.PV_Module_Voltage, dtostrf_buff);
    strcat(measurement_data.PV_Module_Voltage, "V");
    memset(dtostrf_buff, 0, sizeof(dtostrf_buff));
    Serial.println("\n Device connected..\n");
    delay(10);

    pTxCharacteristic->setValue(measurement_data.Data_Block_Caption);
    pTxCharacteristic->notify();
    pTxCharacteristic->setValue(measurement_data.Meas_Box_Temperature);
    pTxCharacteristic->notify();
  }

  if (!deviceConnected && oldDeviceConnected) {
    delay(500);
    pServer->startAdvertising();
    Serial.println("\nStart advertising");
    oldDeviceConnected = deviceConnected;
  }

  if (deviceConnected && !oldDeviceConnected) {
    oldDeviceConnected = deviceConnected;
  }
}

void writeLEDpin() {
  // Implement the function logic here
  if ((ctrl_FLAG_REG & 0x01) == 0x01) {
    ctrl_FLAG_REG = (ctrl_FLAG_REG & (~0x01));
    ctrl_FLAG_REG |= 0x10;
    attachInterrupt(INA_ADC_BUSY_pin, writeLEDpin, RISING);
  }
}
```

6. REFERANCES

1. Introduction The KiCad Team. (n.d.). Available at: https://docs.kicad.org/7.0/en/introduction/introduction.pdf [Accessed 29 Jun. 2023].

2. KiCad The KiCad Team. (n.d.). Available at: https://docs.kicad.org/7.0/en/kicad/kicad.pdf [Accessed 29 Jun. 2023].

3. Schematic Editor The KiCad Team. (n.d.). Available at: https://docs.kicad.org/7.0/en/eeschema/eeschema.pdf [Accessed 29 Jun. 2023].

4. PCB Editor The KiCad Team. (n.d.). Available at: https://docs.kicad.org/7.0/en/pcbnew/pcbnew.pdf [Accessed 29 Jun. 2023].

5. INA239 85-V, 16-Bit, High-Precision Power Monitor With SPI Interface 1 Features. (n.d.). Available at : https://www.ti.com/lit/ds/symlink/ina239.pdf?ts=1687173352403&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FINA239 [Accessed 29 Jun. 2023].

6. Analog.com. (2016). *ADuM120N Datasheet and Product Info | Analog Devices*. [online] Available at: https://www.analog.com/en/products/adum120n.html?doc=ADuM120N_121N.pdf#product-documentation [Accessed 29 Jun. 2023].

7. Analog.com. (2016). *ADuM121N Datasheet and Product Info | Analog Devices*. [online] Available at: https://www.analog.com/en/products/adum121n.html?doc=ADuM120N_121N.pdf#product-documentation [Accessed 29 Jun. 2023].

8. www.espressif.com. (n.d.). *ESP32-S3 Wi-Fi & Bluetooth 5 (LE) MCU | Espressif Systems*. [online] Available at: https://www.espressif.com/en/products/socs/esp32-s3#:~:text=ESP32%2DS3%20is%20a%20dual [Accessed 29 Jun. 2023].