Sri-Link development board Testing

Dona Dilini Manushi

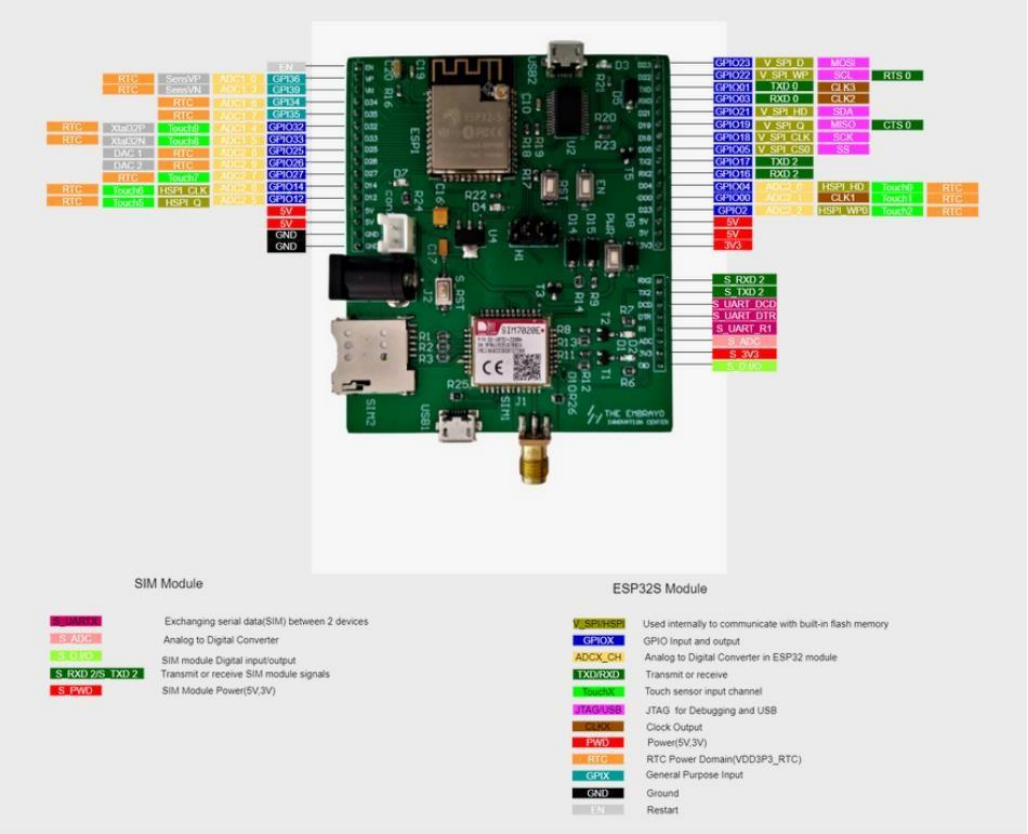


Figure 1: Pinout Diagram of the Sri-Link Board

Soil sensor Testing with the Sri-Link using Arduino IDE

The code for testing the soil sensor was provided by the supervisor and it is written below.

```
#include <SoftwareSerial.h>


#define PIN_RX 32 //RO
#define PIN_TX 14 //DI
// RS485 setup with ESp32
#define RE 26  // Connect RE terminal with 32 of ESP
#define DE 33    // Connect DE terminal with 33 of ESP
#define HARDWARE_SW_1 27

SoftwareSerial mod(PIN_RX,PIN_TX);

const byte test[] = {0x01, 0x03, 0x00, 0x00, 0x00, 0x07, 0x04, 0x08};

byte readings[20];
float avgReading[7];

void setup() {
 Serial.begin(115200);
 enableExternalPrep();
```

```
 initModBus();
 Serial.println("Sensor Reding Started");
}

void initModBus(){
mod.begin(9600);
 pinMode(RE, OUTPUT);
 pinMode(DE, OUTPUT);
 delay(1000);
 readSensor();
 delay(1000);
 readSensor();
 delay(1000);
 readSensor();
 delay(1000);
}

boolean readSensor(){
  digitalWrite(DE,HIGH);
  digitalWrite(RE,HIGH);
  delay(100);
  if(mod.write(test,sizeof(test))==8){
    digitalWrite(DE,LOW);
    digitalWrite(RE,LOW);
    for(byte i=0;i<19;i++){
       readings[i] = mod.read();
    }
  }
  if((readings[0]==1)&&(readings[1]==3)&&(readings[2]==14)){
      return true;
    }
  else{
      return false;
    }
}

void printSensorDataHex(){
   for(byte i=0;i<19;i++){
     Serial.print(readings[i],HEX);
     Serial.print(" ");
    }
    Serial.println("");
}


void printTemp(){
int result = (readings[5]*256) + readings[6];
```

```
Serial.printf("Temp : %f \n",(float(result)/10));
}


void printPH(){
int result = (readings[9]*256) + readings[10];
Serial.printf("PH : %f \n",(float(result)/10));
}

void printMoisture(){
int result = (readings[3]*256) + readings[4];
Serial.printf("Moisture : %f \n",(float(result)/10));
}

void printCon(){
int result = (readings[7]*256) + readings[8];
Serial.printf("Conductivity : %d \n",result);
}

void printN(){
int result = (readings[11]*256) + readings[12];
Serial.printf("N : %f \n",(float(result)/10));
}

void printP(){
int result = (readings[13]*256) + readings[14];
Serial.printf("P : %f \n",(float(result)/10));
}

void printK(){
int result = (readings[15]*256) + readings[16];
Serial.printf("K : %f \n",(float(result)/10));
}

float getTempe(){
int result = (readings[5]*256) + readings[6];
float temp=float(result)/10;
//Serial.printf("Temp : %f \n",temp);
return temp;
}

float getMoisture(){
int result = (readings[3]*256) + readings[4];
float m=float(result)/10;
return m;
}
```

```
float getPH(){
int result = (readings[9]*256) + readings[10];
float ph=float(result)/10;
//Serial.printf("Temp : %f \n",temp);
return ph;
}

float getCon(){
int result = (readings[7]*256) + readings[8];
return float(result);
}

float getN(){
int result = (readings[11]*256) + readings[12];
float n=float(result)/10;
return n;
}

float getP(){
int result = (readings[13]*256) + readings[14];
float p=float(result)/10;
return p;
}

float getK(){
int result = (readings[15]*256) + readings[16];
float k=float(result)/10;
return k;
}


void cleanAvgDataBuff(){
   avgReading[0]=0.0;
   avgReading[1]=0.0;
   avgReading[2]=0.0;
   avgReading[3]=0.0;
   avgReading[4]=0.0;
   avgReading[5]=0.0;
   avgReading[6]=0.0;
}

String getAvgReadings(){

  if(readSensor()){
        for(byte i=0; i<4; i++){
           delay(1000);
           if(readSensor()){
```

```
            avgReading[0]+=getTempe();
            avgReading[1]+=getMoisture();
            avgReading[2]+=getCon();
            avgReading[3]+=getPH();
            avgReading[4]+=getN();
            avgReading[5]+=getP();
            avgReading[6]+=getK();
            }
            else{
              return "";
             }
          }

          String d =String((float(avgReading[0])/4),2)+"
"+String((float(avgReading[1])/4),1)+" "+String((float(avgReading[2])/4),1)+"
"+String((float(avgReading[3])/4),1)+" "+String((float(avgReading[4])/4),1)+"
"+String((float(avgReading[5])/4),1)+" "+String((float(avgReading[6])/4),1);
          cleanAvgDataBuff();
          if(((avgReading[0])/4)<60){
             return d;
           }
          else{
             return "";
           }
     }

  else{
      return "";
    }
}

void enableExternalPrep(){
   pinMode(HARDWARE_SW_1,OUTPUT);
   digitalWrite(HARDWARE_SW_1,HIGH);
   delay(100);
}

void disableExternalPrep(){
   digitalWrite(HARDWARE_SW_1,LOW);
  // gpio_hold_en(GPIO_NUM_27);
}


void loop(){


  //printTemp();
```

```
  /*int count=0;
  while(!readSensor()){
  if(count>10){
    Serial.println("Sensor Error");
     break;
  }

  Serial.println("Sensor Reading....");

  count++;
  delay(1000);
  }*/
  enableExternalPrep();
  String v=getAvgReadings();
  delay(1000);
  digitalWrite(RE,HIGH);
  digitalWrite(DE,LOW);
  delay(1000);
  if(v==""){
    Serial.println("Failed");
    }
  else{
    Serial.println(v);
    }


  delay(5000);
}
```

But we got the message “Failed” on the serial monitor since it didn’t get any readings so I modified the code and modified code is below.

```
#include <SoftwareSerial.h>

#define PIN_RX 16 // RO (Receive Pin)
#define PIN_TX 17 // DI (Transmit Pin)

// RS485 setup with ESP32
#define RE 32 // Connect RE terminal with 32 of ESP (Receiver Enable)
#define DE 33 // Connect DE terminal with 33 of ESP (Driver Enable)
#define HARDWARE_SW_1 27 // Hardware switch control pin

SoftwareSerial mod(PIN_RX, PIN_TX);

const byte test[] = {0x01, 0x03, 0x00, 0x00, 0x00, 0x07, 0x04, 0x08}; // Test
command to send to the sensor

byte readings[20]; // Buffer to store received data
```

```
float avgReading[7]; // Array to store average sensor readings
unsigned long timeoutStartTime; // Declare the timeout start time variable

void setup() {
  Serial.begin(115200); // Initialize serial communication
  enableExternalPrep(); // Enable external preparation
  initModBus(); // Initialize ModBus communication
  Serial.println("Sensor Reading Started");
}

void initModBus() {
  mod.begin(9600); // Initialize SoftwareSerial communication with the sensor
  pinMode(RE, OUTPUT); // Set RE pin as output
  pinMode(DE, OUTPUT); // Set DE pin as output
  digitalWrite(RE, LOW); // Set RE pin low by default
  digitalWrite(DE, LOW); // Set DE pin low by default
  delay(1000); // Wait for a brief moment
}

boolean readSensor() {
  digitalWrite(DE, HIGH); // Enable data transmission
  digitalWrite(RE, HIGH); // Enable data reception
  delay(10); // Short delay for faster response
  timeoutStartTime = millis(); // Record the start time for timeout tracking
  if (mod.write(test, sizeof(test)) == 8) { // Send test command to sensor
    delay(10); // Short delay for faster response
    digitalWrite(DE, LOW); // Disable data transmission
    digitalWrite(RE, LOW); // Disable data reception
    byte bytesRead = 0;
    while (bytesRead < 19 && millis() - timeoutStartTime < 1000) {
      if (mod.available()) {
        readings[bytesRead] = mod.read(); // Read data byte by byte
        bytesRead++;
      }
    }
    return bytesRead == 19; // Return true if all expected bytes were read
  }
  return false; // Return false if communication was unsuccessful
}

void loop() {
  if (readSensor()) {
    printSensorDataHex(); // Debug: Print the received data in hex
    printTemp();
    printPH();
    printMoisture();
    printCon();
```

```
    printN();
    printP();
    printK();
  } else {
    Serial.println("Failed to read sensor data");
  }
  delay(5000); // Delay before next reading
}

void printSensorDataHex() {
  for (byte i = 0; i < 19; i++) {
    Serial.print(readings[i], HEX); // Print received data in hexadecimal format
    Serial.print(" ");
  }
  Serial.println();
}

// Functions to print specific sensor readings
void printTemp() {
  int result = (readings[5] * 256) + readings[6];
  Serial.print("Temp: ");
  Serial.println(float(result) / 10, 1);
}

void printPH() {
  int result = (readings[9] * 256) + readings[10];
  Serial.print("pH: ");
  Serial.println(float(result) / 10, 1);
}

void printMoisture() {
  int result = (readings[3] * 256) + readings[4];
  Serial.print("Moisture: ");
  Serial.println(float(result) / 10, 1);
}

void printCon() {
  int result = (readings[7] * 256) + readings[8];
  Serial.print("Conductivity: ");
  Serial.println(result);
}

void printN() {
  int result = (readings[11] * 256) + readings[12];
  Serial.print("N: ");
  Serial.println(float(result) / 10, 1);
}
```

```
void printP() {
  int result = (readings[13] * 256) + readings[14];
  Serial.print("P: ");
  Serial.println(float(result) / 10, 1);
}

void printK() {
  int result = (readings[15] * 256) + readings[16];
  Serial.print("K: ");
  Serial.println(float(result) / 10, 1);
}

// Enable external preparation
void enableExternalPrep() {
  pinMode(HARDWARE_SW_1, OUTPUT); // Set hardware switch control pin as output
  digitalWrite(HARDWARE_SW_1, HIGH); // Turn on the hardware switch
  delay(100); // Wait for a brief moment
}
```

Reduced Delay in Communication:

In the initial code, there were delays of 100 milliseconds after setting the DE and RE pins high. I reduced these delays to 10 milliseconds in the modified code. This change allows the communication to start faster after setting the pins high, which might help in receiving the sensor data more promptly.

Timeout Mechanism for Reading:

I added a timeout mechanism for reading data from the sensor. In the initial code, there was no mechanism to handle cases where the sensor data might not be received within a reasonable time. In the modified code, I introduced a timeoutStartTime variable to record the start time when data communication is initiated. Then, I used this variable to compare against the current time in a loop to ensure that the communication does not get stuck indefinitely waiting for data. This helps prevent the code from getting stuck in a non-responsive state.

Read Byte-by-Byte:

In the initial code, the entire response from the sensor was expected to be received at once, which might not always be the case due to potential timing mismatches. In the modified code, I changed the approach to read data byte by byte as it becomes available. This ensures that the code can handle variations in the timing of data transmission from the sensor.

Print Received Data:

I added a debug function, printSensorDataHex(), to print the received data in hexadecimal format. This helps in debugging and understanding the raw data received from the sensor.

Print Functions Reordering:

I reordered the print functions for temperature, pH, moisture, conductivity, nitrogen, phosphorus, and potassium readings after the data is read successfully from the sensor. This organization improves the readability of the code and ensures that data is printed in a consistent manner.

Initialization Sequence Change:

I removed the repeated calls to readSensor() after delays in the initModBus() function. This is because the readSensor() function is meant to read actual sensor data, not just perform communication initialization. The repeated calls to readSensor() were not necessary and could potentially interfere with proper initialization.

RE and DE Pin Configuration:

I set the RE and DE pins low by default in the initModBus() function. This ensures that the pins start in the correct state before communication begins.

Avoiding Long Delays:

I removed the long delays that were present in the initial code. Long delays can slow down the overall operation of the code and potentially lead to missed sensor data.

By making these changes, the modified code aims to provide a more robust and responsive communication protocol with the sensor. It incorporates timeout handling to prevent the code from getting stuck and utilizes byte-by-byte data reception for flexibility. The changes should contribute to more reliable data acquisition from the sensor.

As for why the modified code worked while the initial code didn't, the modifications were focused on improving the timing, handling communication delays, and implementing a timeout mechanism. These changes collectively enhance the code's ability to handle variations in the timing of data transmission from the sensor and ensure that the code doesn't get stuck in a non-responsive state. This can result in improved reliability and consistent data reading from the sensor.

The main modification was made in the readSensor function, and I introduced the new variable unsigned long timeoutStartTime for timeout tracking.

```
boolean readSensor() {
  digitalWrite(DE, HIGH);
  digitalWrite(RE, HIGH);
  delay(10);
  timeoutStartTime = millis(); // * Record the start time for timeout tracking
  if (mod.write(test, sizeof(test)) == 8) {
    delay(10);
    digitalWrite(DE, LOW);
    digitalWrite(RE, LOW);
    byte bytesRead = 0;
    while (bytesRead < 19 && millis() - timeoutStartTime < 1000) { // * Introduce timeout mechanism
      if (mod.available()) {
        readings[bytesRead] = mod.read();
        bytesRead++;
      }
    }
    return bytesRead == 19; // * Return true if all expected bytes were read
  }
  return false;
}
```

Timeout Tracking (timeoutStartTime): I introduced a new variable called timeoutStartTime, which is of type unsigned long. This variable is used to keep track of the start time when the communication with the sensor begins. This is essential to implement a timeout mechanism. If the communication with the sensor takes too long (more than a specified time), we want to exit the communication loop to avoid hanging indefinitely.

Timeout Mechanism (millis()): I added a check within the while loop that reads data from the sensor. This check involves comparing the difference between the current time (obtained using millis()) and the timeoutStartTime. If this difference exceeds a certain threshold (in this case, 1000 milliseconds or 1 second), the loop exits, even if not all expected bytes were received. This prevents the program from getting stuck in a communication loop in case the sensor does not respond as expected.

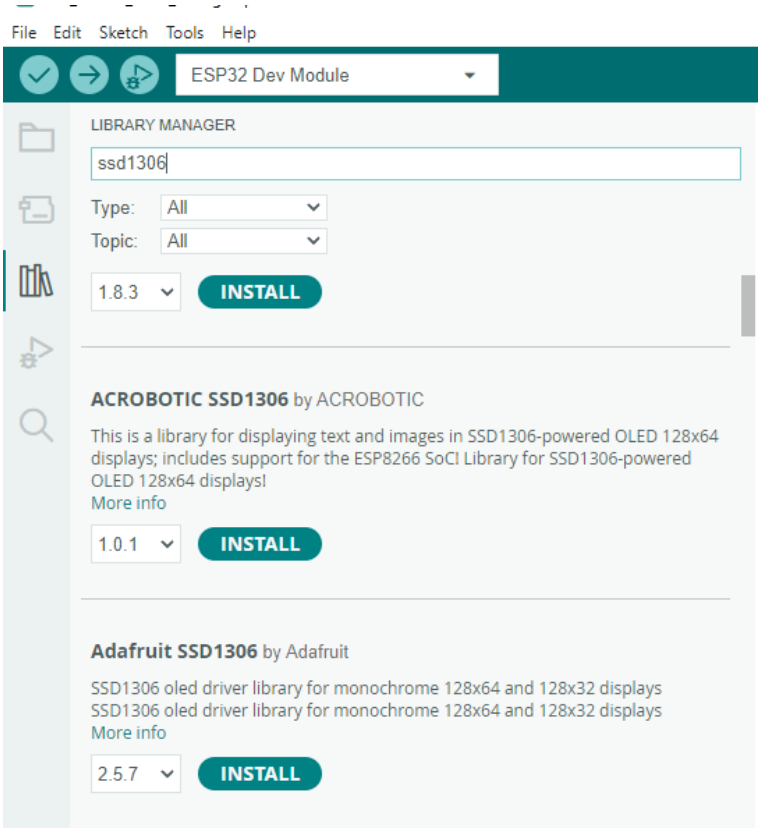
Return Condition: The return condition within the readSensor function has also been modified. Previously, it returned true only if the exact number of bytes expected (19) were read from the sensor. Now, it returns true if all expected bytes were read within the specified timeout period.

By introducing these changes, I aimed to make the readSensor function more robust and responsive. It prevents the program from getting stuck if the sensor communication takes too long, and it provides better control over handling potential communication issues.

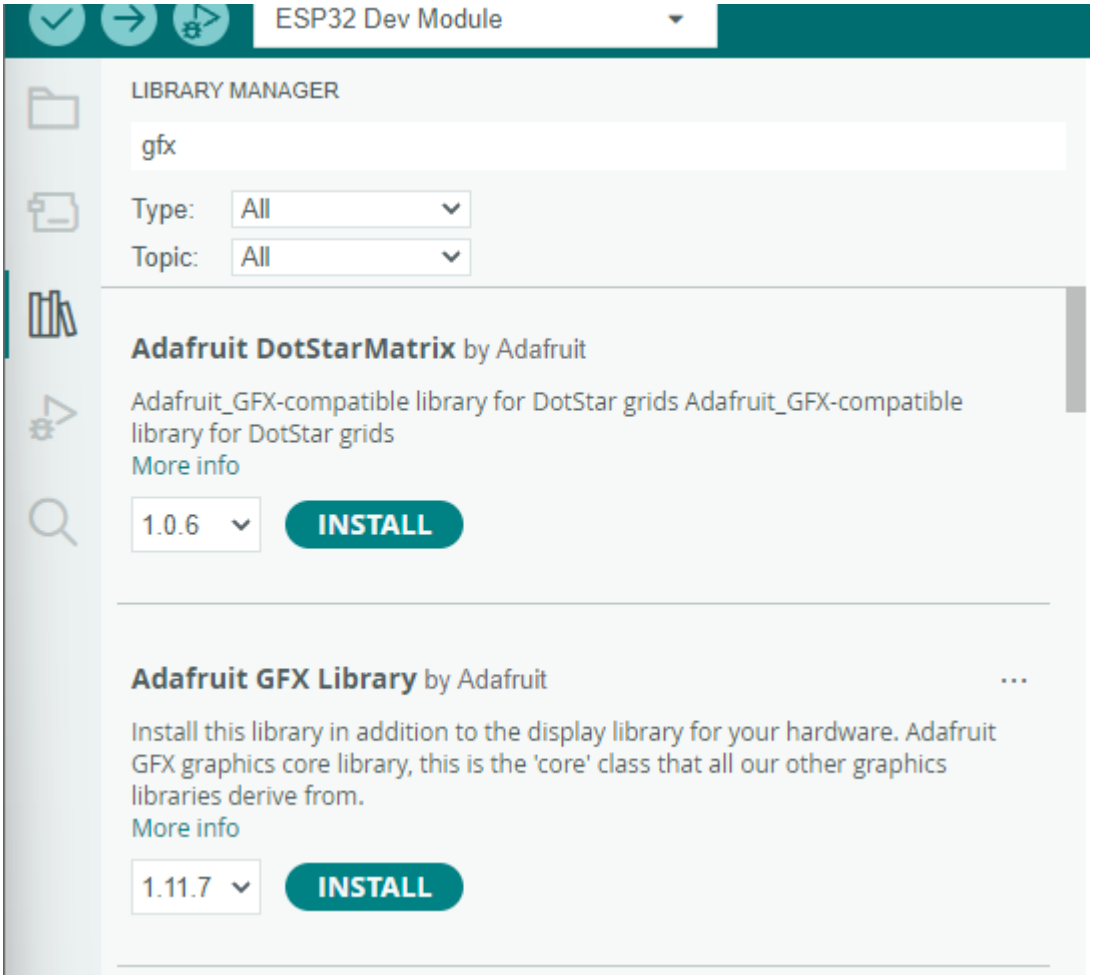
OLED display testing with Sri-Link using Arduino IDE

I opened Arduino IDE and went to Sketch > Include Library > Manage Libraries. The Library Manager opened

I typed “SSD1306” in the search box and installed the SSD1306 library from Adafruit.

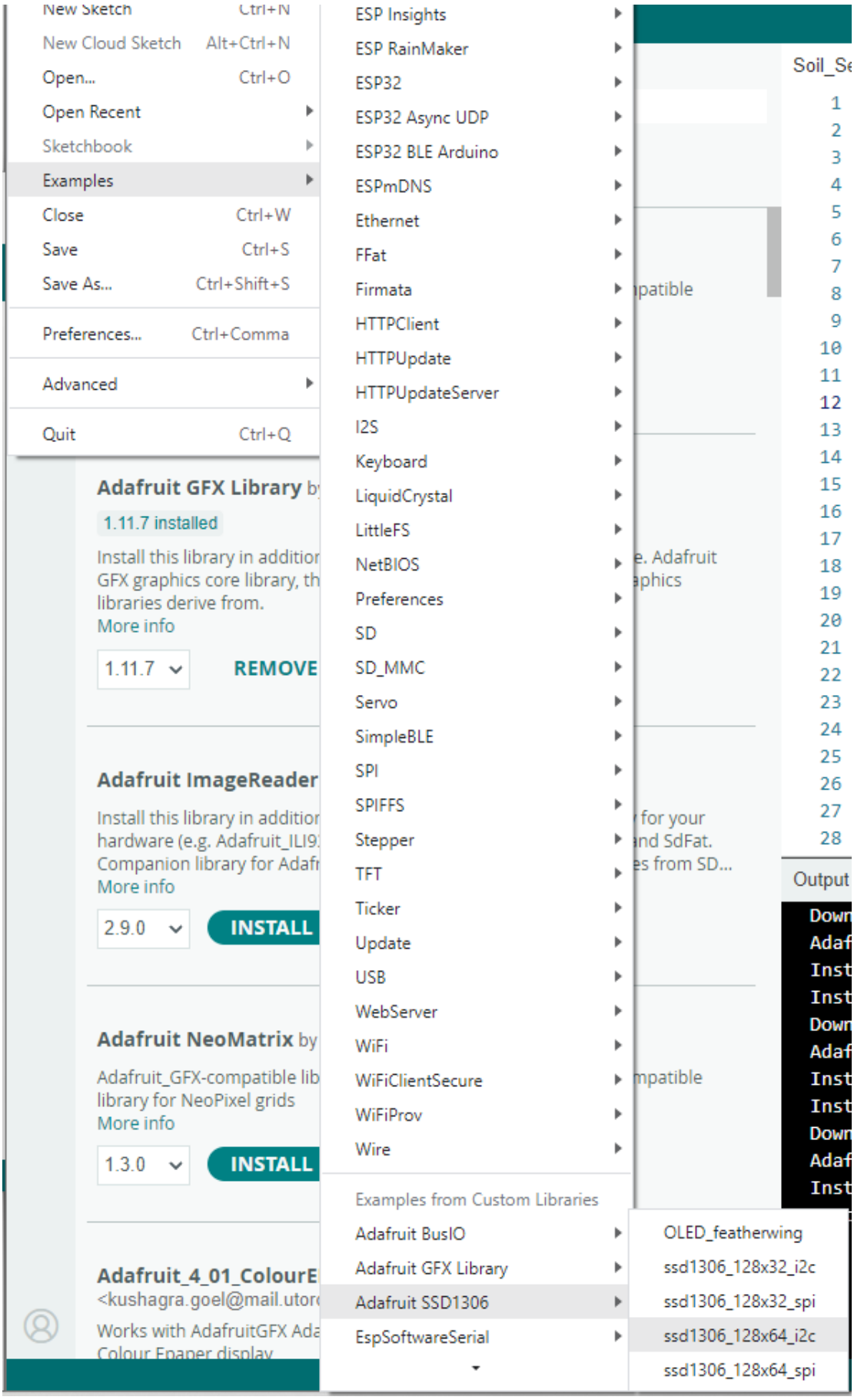


After installing the SSD1306 library from Adafruit, I typed “GFX” in the search box and installed the library.



After wiring the OLED display to the Sri-Link and installing all required libraries, I used one example from the library to see if everything is working properly.

In Arduino IDE, I went to File > Examples > Adafruit SSD1306 and selected the example for the I am using.


New Sketch Ctrl+N
New Cloud Sketch Alt+Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Preferences... Ctrl+Comma
Advanced
Quit Ctrl+Q
ESP Insights
ESP RainMaker
ESP32
ESP32 Async UDP
ESP32 BLE Arduino
ESPmDNS
Ethernet
FFat
Firmata
HTTPClient
HTTPUpdate
HTTPUpdateServer
I2S
Keyboard
LiquidCrystal
LittleFS
NetBIOS
Preferences
SD
SD_MMC
Servo
SimpleBLE
SPI
SPIFFS
Stepper
TFT
Ticker
Update
USB
WebServer
WiFi
WiFiClientSecure
WiFiProv
Wire
Examples from Custom Libraries
Adafruit BusIO
Adafruit GFX Library
Adafruit SSD1306
EspSoftwareSerial
OLED_featherwing
ssd1306_128x32_i2c
ssd1306_128x32_spi
ssd1306_128x64_i2c
ssd1306_128x64_spi
Adafruit GFX Library by
1.11.7 installed
Install this library in addition
GFX graphics core library, th
libraries derive from.
More info
1.11.7
REMOVE
Adafruit ImageReader
Install this library in addition
hardware (e.g. Adafruit_ILI9
Companion library for Adafr
More info
2.9.0
INSTALL
Adafruit NeoMatrix by
Adafruit_GFX-compatible lib
library for NeoPixel grids
More info
1.3.0
INSTALL
Adafruit_4_01_ColourE
<kushagra.goel@mail.utor
Works with AdafruitGFX Ada
Colour Epaper display
Soil_Se
Output
Down
Adaf
Inst
Inst
Down
Adaf
Inst
Inst
Down
Adaf
Inst

The code is below.

```
/*************************************************************************
 This is an example for our Monochrome OLEDs based on SSD1306 drivers

 Pick one up today in the adafruit shop!
 ------> http://www.adafruit.com/category/63_98

 This example is for a 128x32 pixel display using I2C to communicate
 3 pins are required to interface (two I2C and one reset).

 Adafruit invests time and resources providing this open
 source code, please support Adafruit and open-source
 hardware by purchasing products from Adafruit!

 Written by Limor Fried/Ladyada for Adafruit Industries,
 with contributions from the open source community.
 BSD license, check license.txt for more information
 All text above, and the splash screen below must be
 included in any redistribution.
 *************************************************************************/

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
// The pins for I2C are defined by the Wire-library.
// On an arduino UNO:       A4(SDA), A5(SCL)
// On an arduino MEGA 2560: 20(SDA), 21(SCL)
// On an arduino LEONARDO:   2(SDA),  3(SCL), ...
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C
for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define NUMFLAKES     10 // Number of snowflakes in the animation example

#define LOGO_HEIGHT   16
#define LOGO_WIDTH    16
static const unsigned char PROGMEM logo_bmp[] =
{ 0b00000000, 0b11000000,
```

```
  0b00000001, 0b11000000,
  0b00000001, 0b11000000,
  0b00000011, 0b11100000,
  0b11110011, 0b11100000,
  0b11111110, 0b11111000,
  0b01111110, 0b11111111,
  0b00110011, 0b10011111,
  0b00011111, 0b11111100,
  0b00001101, 0b01110000,
  0b00011011, 0b10100000,
  0b00111111, 0b11100000,
  0b00111111, 0b11110000,
  0b01111100, 0b11110000,
  0b01110000, 0b01110000,
  0b00000000, 0b00110000 };

void setup() {
  Serial.begin(9600);

  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  display.display();
  delay(2000); // Pause for 2 seconds

  // Clear the buffer
  display.clearDisplay();

  // Draw a single pixel in white
  display.drawPixel(10, 10, SSD1306_WHITE);

  // Show the display buffer on the screen. You MUST call display() after
  // drawing commands to make them visible on screen!
  display.display();
  delay(2000);
  // display.display() is NOT necessary after every single drawing command,
  // unless that's what you want...rather, you can batch up a bunch of
  // drawing operations and then update the screen all at once by calling
  // display.display(). These examples demonstrate both approaches...

  testdrawline();      // Draw many lines
```

```
  testdrawrect();      // Draw rectangles (outlines)

  testfillrect();      // Draw rectangles (filled)

  testdrawcircle();    // Draw circles (outlines)

  testfillcircle();    // Draw circles (filled)

  testdrawroundrect(); // Draw rounded rectangles (outlines)

  testfillroundrect(); // Draw rounded rectangles (filled)

  testdrawtriangle();  // Draw triangles (outlines)

  testfilltriangle();  // Draw triangles (filled)

  testdrawchar();      // Draw characters of the default font

  testdrawstyles();    // Draw 'stylized' characters

  testscrolltext();    // Draw scrolling text

  testdrawbitmap();    // Draw a small bitmap image

  // Invert and restore display, pausing in-between
  display.invertDisplay(true);
  delay(1000);
  display.invertDisplay(false);
  delay(1000);

  testanimate(logo_bmp, LOGO_WIDTH, LOGO_HEIGHT); // Animate bitmaps
}

void loop() {
}

void testdrawline() {
  int16_t i;

  display.clearDisplay(); // Clear display buffer

  for(i=0; i<display.width(); i+=4) {
    display.drawLine(0, 0, i, display.height()-1, SSD1306_WHITE);
    display.display(); // Update screen with each newly-drawn line
    delay(1);
  }
  for(i=0; i<display.height(); i+=4) {
```

```
  display.drawLine(0, 0, display.width()-1, i, SSD1306_WHITE);
  display.display();
  delay(1);
}
delay(250);

display.clearDisplay();

for(i=0; i<display.width(); i+=4) {
  display.drawLine(0, display.height()-1, i, 0, SSD1306_WHITE);
  display.display();
  delay(1);
}
for(i=display.height()-1; i>=0; i-=4) {
  display.drawLine(0, display.height()-1, display.width()-1, i, SSD1306_WHITE);
  display.display();
  delay(1);
}
delay(250);

display.clearDisplay();

for(i=display.width()-1; i>=0; i-=4) {
  display.drawLine(display.width()-1, display.height()-1, i, 0, SSD1306_WHITE);
  display.display();
  delay(1);
}
for(i=display.height()-1; i>=0; i-=4) {
  display.drawLine(display.width()-1, display.height()-1, 0, i, SSD1306_WHITE);
  display.display();
  delay(1);
}
delay(250);

display.clearDisplay();

for(i=0; i<display.height(); i+=4) {
  display.drawLine(display.width()-1, 0, 0, i, SSD1306_WHITE);
  display.display();
  delay(1);
}
for(i=0; i<display.width(); i+=4) {
  display.drawLine(display.width()-1, 0, i, display.height()-1, SSD1306_WHITE);
  display.display();
  delay(1);
}
```

```
  delay(2000); // Pause for 2 seconds
}

void testdrawrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2; i+=2) {
    display.drawRect(i, i, display.width()-2*i, display.height()-2*i,
SSD1306_WHITE);
    display.display(); // Update screen with each newly-drawn rectangle
    delay(1);
  }

  delay(2000);
}

void testfillrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2; i+=3) {
    // The INVERSE color is used so rectangles alternate white/black
    display.fillRect(i, i, display.width()-i*2, display.height()-i*2,
SSD1306_INVERSE);
    display.display(); // Update screen with each newly-drawn rectangle
    delay(1);
  }

  delay(2000);
}

void testdrawcircle(void) {
  display.clearDisplay();

  for(int16_t i=0; i<max(display.width(),display.height())/2; i+=2) {
    display.drawCircle(display.width()/2, display.height()/2, i, SSD1306_WHITE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testfillcircle(void) {
  display.clearDisplay();

  for(int16_t i=max(display.width(),display.height())/2; i>0; i-=3) {
    // The INVERSE color is used so circles alternate white/black
```

```
    display.fillCircle(display.width() / 2, display.height() / 2, i,
SSD1306_INVERSE);
    display.display(); // Update screen with each newly-drawn circle
    delay(1);
  }

  delay(2000);
}

void testdrawroundrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2-2; i+=2) {
    display.drawRoundRect(i, i, display.width()-2*i, display.height()-2*i,
      display.height()/4, SSD1306_WHITE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testfillroundrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2-2; i+=2) {
    // The INVERSE color is used so round-rects alternate white/black
    display.fillRoundRect(i, i, display.width()-2*i, display.height()-2*i,
      display.height()/4, SSD1306_INVERSE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testdrawtriangle(void) {
  display.clearDisplay();

  for(int16_t i=0; i<max(display.width(),display.height())/2; i+=5) {
    display.drawTriangle(
      display.width()/2  , display.height()/2-i,
      display.width()/2-i, display.height()/2+i,
      display.width()/2+i, display.height()/2+i, SSD1306_WHITE);
    display.display();
    delay(1);
  }
```

```
  delay(2000);
}

void testfilltriangle(void) {
  display.clearDisplay();

  for(int16_t i=max(display.width(),display.height())/2; i>0; i-=5) {
    // The INVERSE color is used so triangles alternate white/black
    display.fillTriangle(
      display.width()/2  , display.height()/2-i,
      display.width()/2-i, display.height()/2+i,
      display.width()/2+i, display.height()/2+i, SSD1306_INVERSE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testdrawchar(void) {
  display.clearDisplay();

  display.setTextSize(1);      // Normal 1:1 pixel scale
  display.setTextColor(SSD1306_WHITE); // Draw white text
  display.setCursor(0, 0);     // Start at top-left corner
  display.cp437(true);         // Use full 256 char 'Code Page 437' font

  // Not all the characters will fit on the display. This is normal.
  // Library will draw what it can and the rest will be clipped.
  for(int16_t i=0; i<256; i++) {
    if(i == '\n') display.write(' ');
    else          display.write(i);
  }

  display.display();
  delay(2000);
}

void testdrawstyles(void) {
  display.clearDisplay();

  display.setTextSize(1);             // Normal 1:1 pixel scale
  display.setTextColor(SSD1306_WHITE);        // Draw white text
  display.setCursor(0,0);             // Start at top-left corner
  display.println(F("Hello, world!"));
```

```
  display.setTextColor(SSD1306_BLACK, SSD1306_WHITE); // Draw 'inverse' text
  display.println(3.141592);

  display.setTextSize(2);             // Draw 2X-scale text
  display.setTextColor(SSD1306_WHITE);
  display.print(F("0x")); display.println(0xDEADBEEF, HEX);

  display.display();
  delay(2000);
}

void testscrolltext(void) {
  display.clearDisplay();

  display.setTextSize(2); // Draw 2X-scale text
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(10, 0);
  display.println(F("scroll"));
  display.display();      // Show initial text
  delay(100);

  // Scroll in various directions, pausing in-between:
  display.startscrollright(0x00, 0x0F);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrollleft(0x00, 0x0F);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrolldiagright(0x00, 0x07);
  delay(2000);
  display.startscrolldiagleft(0x00, 0x07);
  delay(2000);
  display.stopscroll();
  delay(1000);
}

void testdrawbitmap(void) {
  display.clearDisplay();

  display.drawBitmap(
    (display.width()  - LOGO_WIDTH ) / 2,
    (display.height() - LOGO_HEIGHT) / 2,
    logo_bmp, LOGO_WIDTH, LOGO_HEIGHT, 1);
  display.display();
  delay(1000);
```

```
}

#define XPOS   0 // Indexes into the 'icons' array in function below
#define YPOS   1
#define DELTAY 2

void testanimate(const uint8_t *bitmap, uint8_t w, uint8_t h) {
  int8_t f, icons[NUMFLAKES][3];

  // Initialize 'snowflake' positions
  for(f=0; f< NUMFLAKES; f++) {
    icons[f][XPOS]   = random(1 - LOGO_WIDTH, display.width());
    icons[f][YPOS]   = -LOGO_HEIGHT;
    icons[f][DELTAY] = random(1, 6);
    Serial.print(F("x: "));
    Serial.print(icons[f][XPOS], DEC);
    Serial.print(F(" y: "));
    Serial.print(icons[f][YPOS], DEC);
    Serial.print(F(" dy: "));
    Serial.println(icons[f][DELTAY], DEC);
  }

  for(;;) { // Loop forever...
    display.clearDisplay(); // Clear the display buffer

    // Draw each snowflake:
    for(f=0; f< NUMFLAKES; f++) {
      display.drawBitmap(icons[f][XPOS], icons[f][YPOS], bitmap, w, h,
SSD1306_WHITE);
    }

    display.display(); // Show the display buffer on the screen
    delay(200);        // Pause for 1/10 second

    // Then update coordinates of each flake...
    for(f=0; f< NUMFLAKES; f++) {
      icons[f][YPOS] += icons[f][DELTAY];
      // If snowflake is off the bottom of the screen...
      if (icons[f][YPOS] >= display.height()) {
        // Reinitialize to a random position, just off the top
        icons[f][XPOS]   = random(1 - LOGO_WIDTH, display.width());
        icons[f][YPOS]   = -LOGO_HEIGHT;
        icons[f][DELTAY] = random(1, 6);
      }
    }
  }
}
```

Testing the OLED display & RTC with the Sri-Link using Arduino IDE.

The code for testing the OLED display & RTC was obtained from the internet. https://microcontrollerslab.com/esp32-ds3231-real-time-clock-rtc-oled/

```
#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>

#include "RTClib.h"


RTC_DS3231 rtc;


char days[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};


Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire, -1);


void setup() {
  Serial.begin(115200);


  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);


  if (! rtc.begin()) {
    Serial.println("Could not find RTC! Check circuit.");
```

```
    while (1);
  }

  rtc.adjust(DateTime(__DATE__, __TIME__));
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(2);
  display.setCursor(0, 20);
  display.print("RTC CLOCK");
  display.display();
  delay(5000);
}

void loop() {
  DateTime now = rtc.now();

  display.clearDisplay();
  display.setTextSize(2);
  display.setCursor(75, 0);
  display.println(now.second(), DEC);
  display.setTextSize(2);
  display.setCursor(25, 0);
  display.println(":");
```

```
display.setTextSize(2);
display.setCursor(65, 0);
display.println(":");
display.setTextSize(2);
display.setCursor(40, 0);
display.println(now.minute(), DEC);
display.setTextSize(2);
display.setCursor(0, 0);
display.println(now.hour(), DEC);
display.setTextSize(2);
display.setCursor(0, 25);
display.println(now.day(), DEC);
display.print(days[now.dayOfTheWeek()]);
display.setTextSize(2);
display.setCursor(20, 25);
display.println("-");
display.setTextSize(2);
display.setCursor(35, 25);
display.println(now.month(), DEC);
display.setTextSize(2);
display.setCursor(60, 25);
display.println("-");
display.setTextSize(2);
```

```
  display.setCursor(75, 25);
  display.println(now.year(), DEC);
  display.display();
}
```