Faculty of Engineering & Applied Science

Smart Device Prototype

Task 2

First Name	Last Name	Student ID
Abdul	Bhutta	100785884

Table of Contents

I. Introduction

This report is an abstract on designing a smart device to capture the real-time image while recording the temperature and humidity at that instant moment using an independent sampling rate. The image will be stored on the SD card while the data for the temperature, humidity, and image number will be stored in a file and saved on the SD card. The report explains the components required and descriptions for each device. The actual image is shown with the pin layouts in the report to help us understand and visualize each component. A final schematic design is shown in detail on how to connect each wire and device to build the prototype. The software used is Arduino IDE to write the code and upload it to the ESP32-CAM device where the microcontroller will run the program. The language used to write the code can be Java or Python. In the report, there are multiple screenshots to show how to install and implement different functions of the application, where each method is described in detail. Please note this is a high-level abstraction with pseudocode and an explanation of how to implement the smart device to capture the current image, temperature, and humidity. Once the actual design is implemented and tested, it can be modified to save the data on a webserver using WIFI and web server libraries. A raspberry pi 3 can also be implemented to store the data/images and act as a web server rather than using the SD card.

II. Components Required

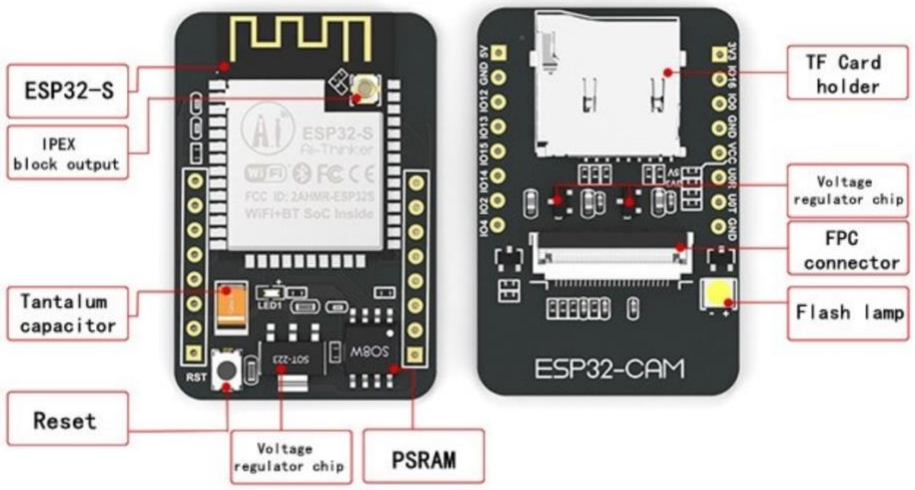
The required items to construct the smart device are shown in the list below with an explanation of what each component is responsible for and the pin layout. The breadboard and wires are used to connect all the components for testing and debugging.

1. Breadboard (Constructing our prototype and testing our smart device)
2. Arduino IDE Software
3. ESP32-CAM with OV2640 Camera
4. FTDI Programmer (ESP32-CAM to USB-to-TTL)
5. DHT11 Sensor
6. 4.7kOhms or 10kOhms Resistor (comes with DHT11)
7. Wires

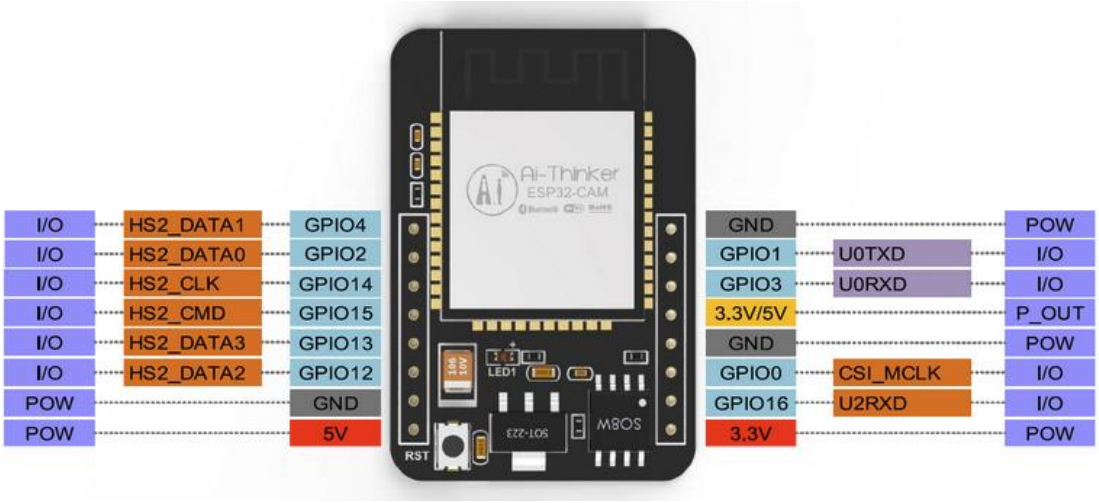
ESP32-CAM

The ESP32-CAM microcontroller comes with a camera module to capture images and store them on an SD card. The microcontroller can be used to connect various other peripherals such as sensors. In the design, we will use the DHT11 sensor and connect it with the ESP32-CAM. The camera will be inserted into the FPC connector. The image below shows an overall description of each connection on the front and back of the microcontroller.

Front and Back

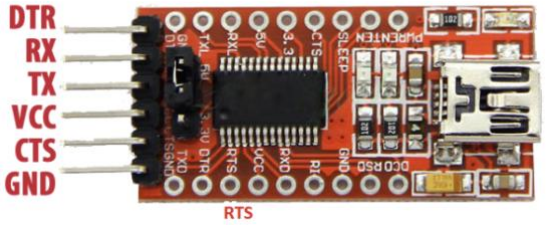


Back with pin connection



FTDI Programmer (USB to TTL)

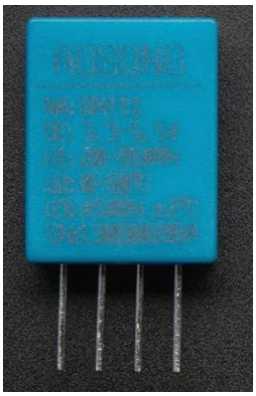
The FTDI programmer is used to upload the code to the ESP32-CAM as the microcontroller does not come with USB connectors. The FTDI pin layout is shown below and what each connector is used for.



Pin	Description
DTR	Reset the hardware device
RX	Receive serial data
TX	Transmit serial data
VCC	Provides 5V/3V
CTS	Enable/Disable programming mode
GND	Ground

DHT11 Sensor

The DHT11 sensor is used to capture the temperature and humidity in the surrounding area every 2 seconds. The pin layout is shown below starting from pin 1 from the far left.

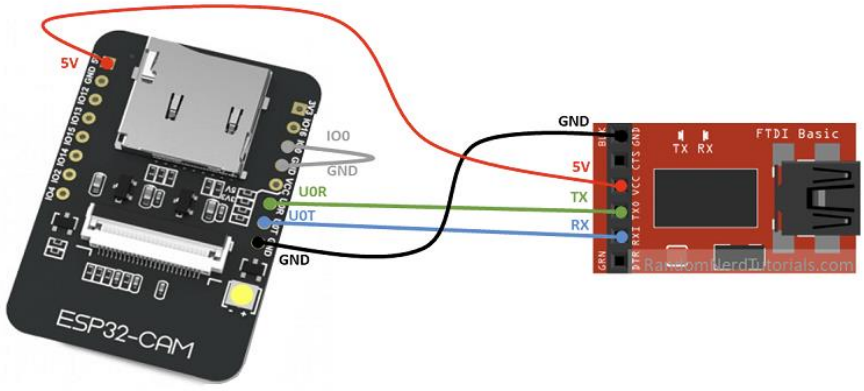


Pin	Description
Pin 1	VCC (5V)
Pin 2	Data out
Pin 3	Not used
Pin 4	Ground

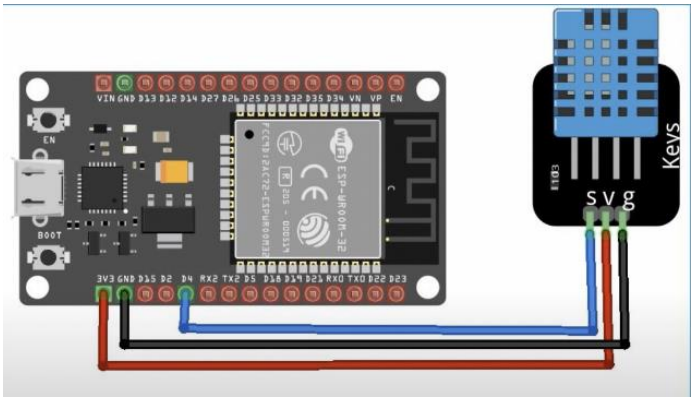
III. Connecting Devices to ESP32-CAM

An illustrative diagram with an actual image of the devices connected is shown below.

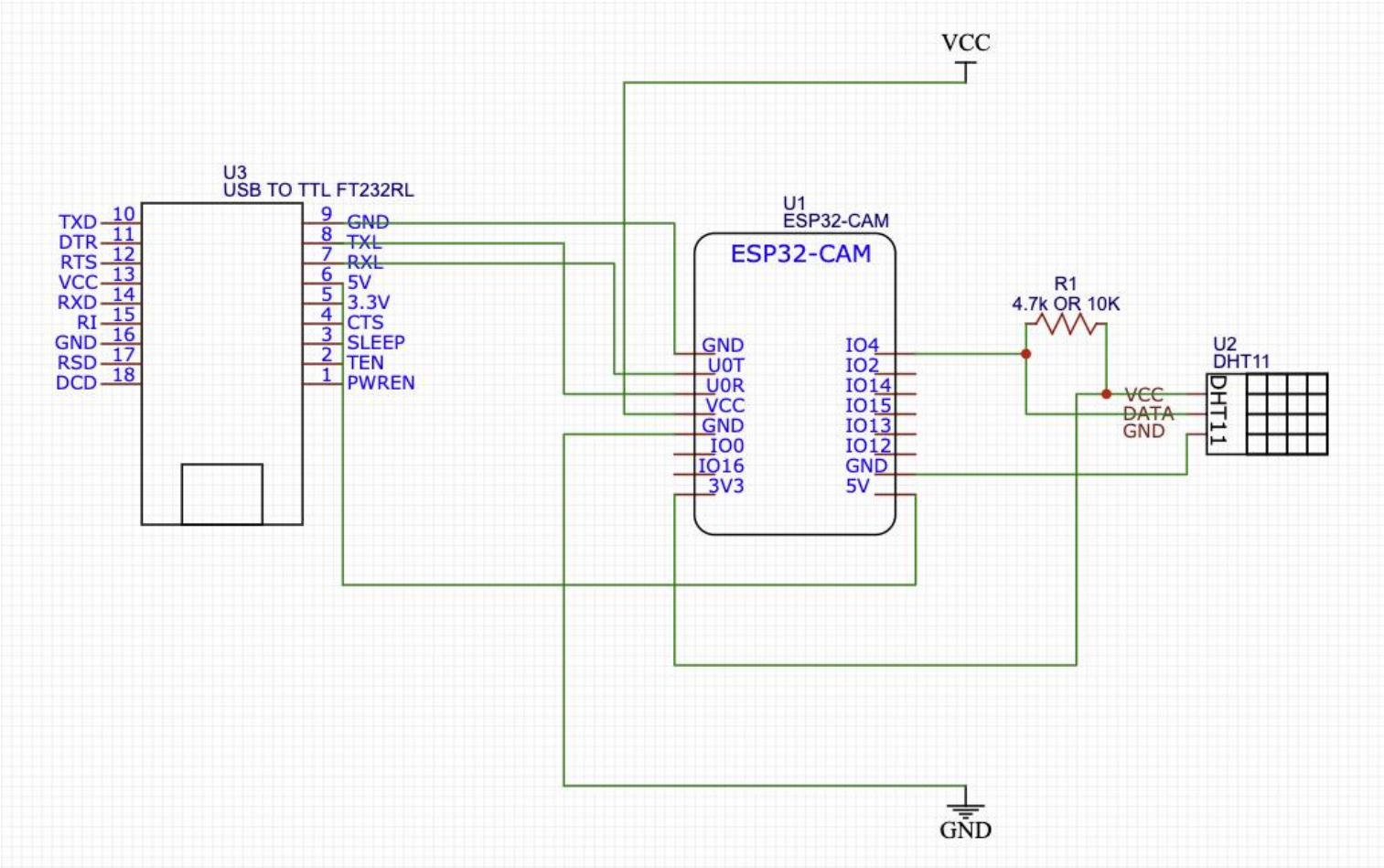
A. Connecting FTDI Programmer to ESP32-CAM



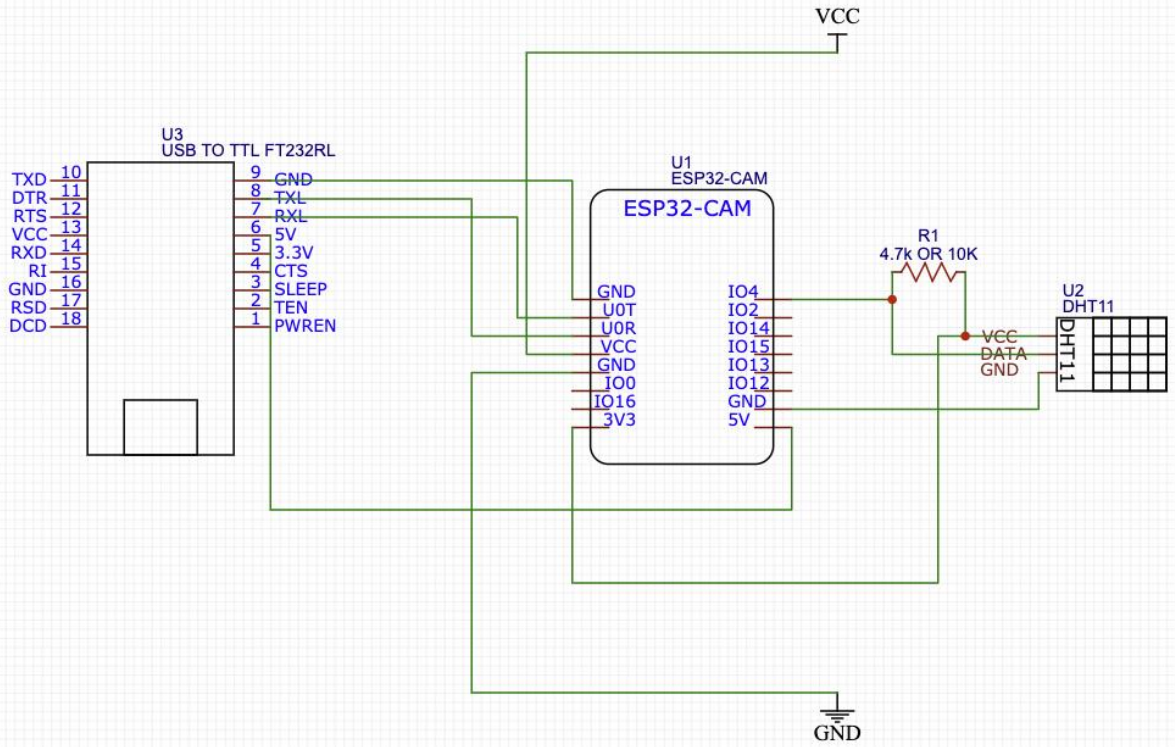
B. Connecting DT11 to ESP32-CAM



IV. Schematic



NOTE: *Connect a 4.7/10K resistor to act as a pull-up resistor on the data line as shown in the schematic diagram above. A Pull-up resistor is used to ensure that pin 2 is in a high state (5V) while using a low amount of current.*



TITLE: Smart Device Prototype		REV: 1.0
	Company: Ontario Tech University	Sheet: 1/1
	Date: 2022-05-25 Drawn By: Abdul Bhutta	

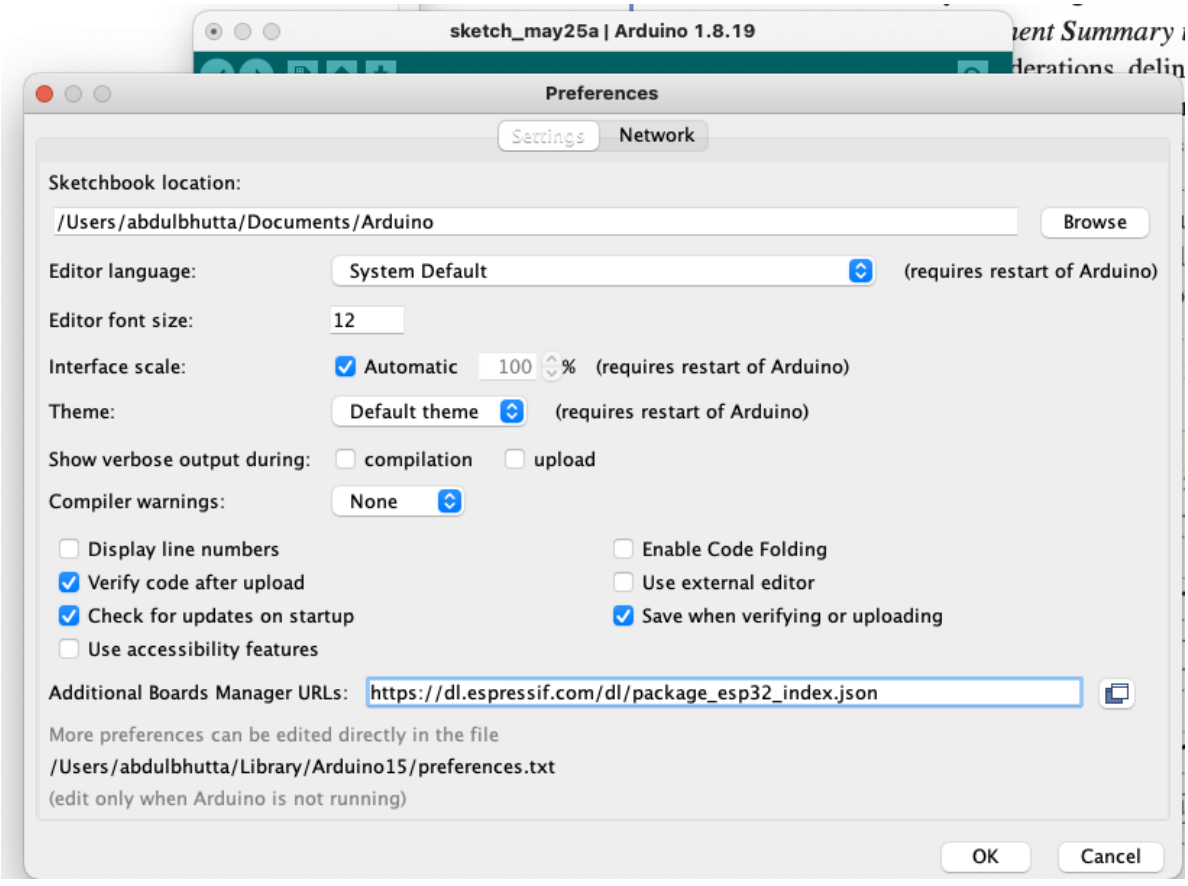
V. Arduino IDE

Arduino IDE software is used to connect to the Arduino hardware and allows the programmer to upload its scripts/program to the device. The installation guide to setting up the ESP32-CAM board and DHT11 sensor library is shown below. All the screenshots taken below were from my current workstation.

Step 1: Download and install the latest Arduino IDE software.

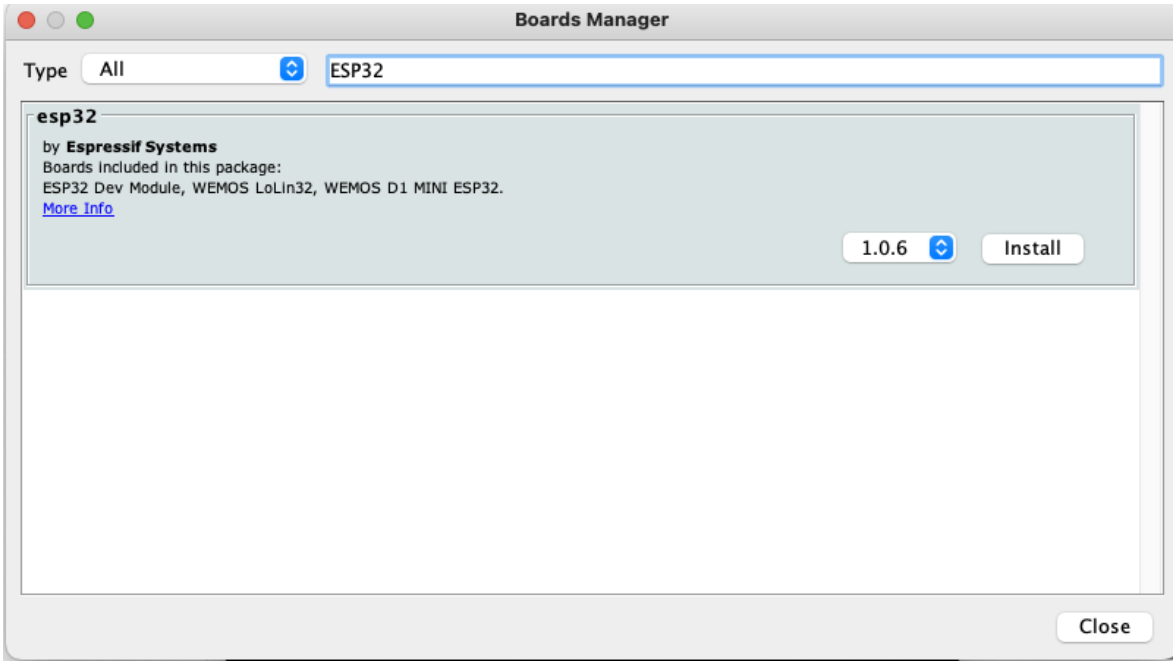
Step 2: Add ESP32 Arduino Library using the URL shown below which contains the board manager for ESP32-CAM

URL: https://dl.espressif.com/dl/package_esp32_index.json

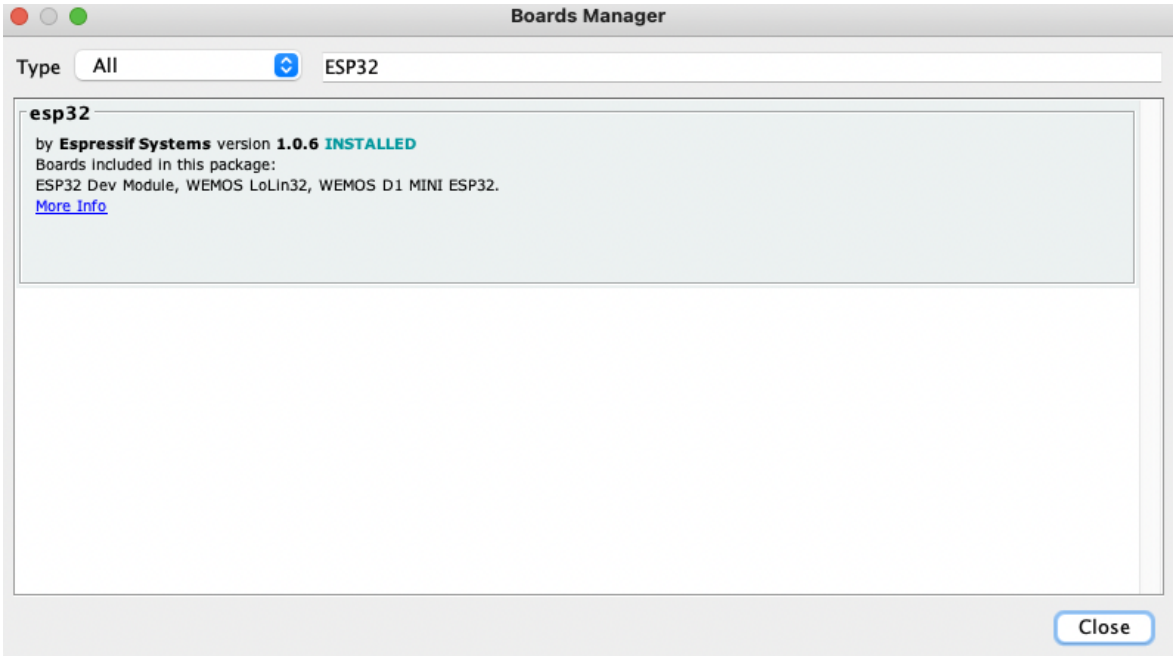


Step 3: Install the ESP32 board through the board manager

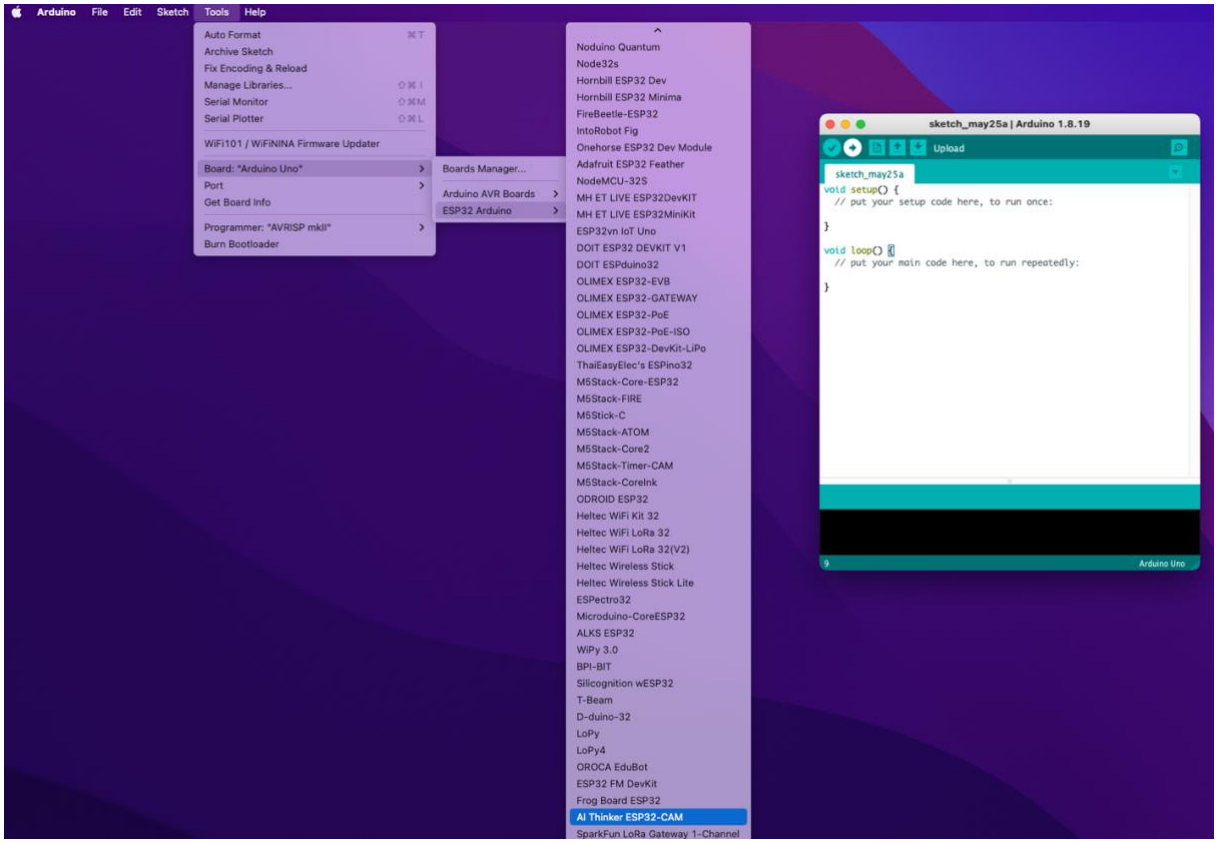
Go to tools -> board manager -> search for ESP32 and click install



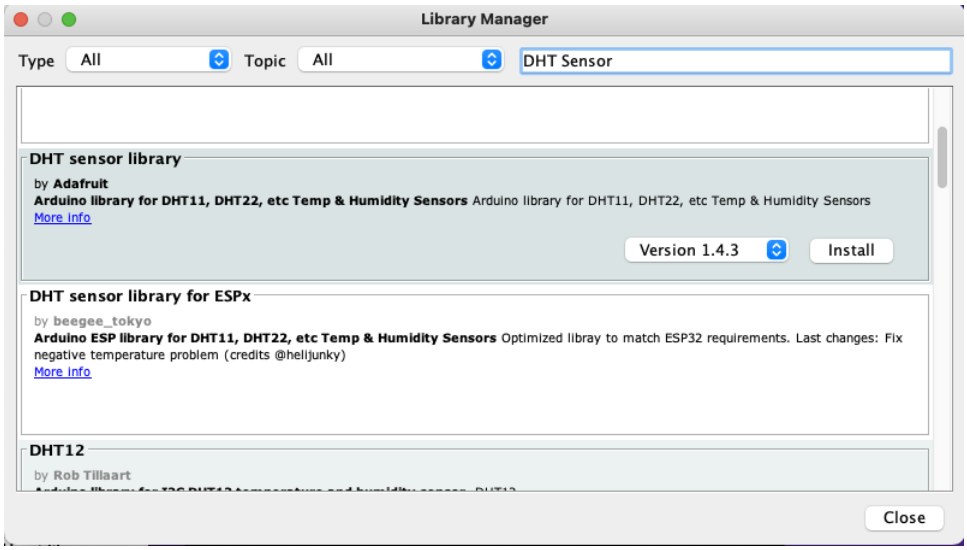
Once the installation is finished, it will show installed as shown below.



Step 4: Choose the AI Thinker ESP32-CAM board



Step 5: Install the DHT11 Library



VI. Pseudocode

This section explains the pseudocode written for the application. Please note the code was not tested nor checked for syntax as it is only for design purposes. The first task is to import all the libraries such as Wi-Fi Library/Webserver (if using the application on a webserver) and the DHT library. The code was designed to be readable and run efficiently while allowing the developer to implement the pseudocode in various languages. Each method in the pseudocode is used to perform one task to allow for high cohesion and low coupling.

Method Name	Functionality
Setup()	The method will be only run once at the start of the application to initialize all the variables/files. The USB port, DHT11 sensor, camera, and SD card will be initialized at the start each time. A new file (if not already created) will be created and named data.txt to capture the readings from the sensors and the image number.
readSensor()	The method will record the current temperature and humidity to the variable's temperature and humidity.
capture()	The method is used to capture the current image from the camera. If the SD card is full, the application will exit and display the error message on the screen. If the SD card is not full, then the application will keep running. It will capture a picture and save the current image in the folder /picture while naming the image 1. Once the image is saved to the path, it will increment the image counter until the SD card is full.
saveData()	The method will append the current saved values in the variable temperature, humidity, and image to the file data.txt.
loop()	The method will keep looping until the application is stopped or it exits. Initially it will delay the data captured by the samplingRate (5000 microseconds = 5 seconds). Every 5 seconds, it will read in the sensor values, capture the current picture, and save the data to a file.

Pseudocode on Arduino IDE

SmartDevicePrototype

```
#include "DHT.h"

//Define sensor type and what pin its connected to
#define DHTTYPE DHT11
#define DHTPIN 2
//Delay is measured in ms, 5000 = 5 seconds
#define SamplingRate = 5000

//Initialize sensor
DHT dht (DHTPIN, DHTTYPE)

//Initialize variables
double temperature=0;
double humidity=0;
int image = 1;

void setup() {
  //Start the usb port and DHT11 sensor
  Serial.begin(115200)
  dht.begin();

  //initialize Camera and SD Card
  esp_err_t err = esp_camera_init(&config);
  SD_MMC.begin();

   //Create new file data.txt to save the captured data
   File data = SD.open("/data.txt");
   writeFile(data, "/data.txt", "Reading Temperature, Humidity, and image every 5 seconds!\n\n");
}

void loop() {
  // put your main code here, to run repeatedly:
  delay(SamplingRate); //Wait for 5 seconds
  readSensor();
  capture();
  saveData();
}

void readSensor() {
  //Read the temperature and humidity
  temperature = dht.readTemperature();
  humidity = dht.readHumidity();
}
```

```
void capture(){
  //If SD card is not full then capture the image and save it!
  if (SDCard != FULL){
    //Take picture with camera
    camera *picture = NULL;
    picture = esp_camera_fb_get();

    //save picture to SD Card
    String Path = "/Picture" + image + ".jpg"; //Path to save the image in picture folder

    //Save the image to SD Card
    fs::FS &fs = SD_MMC;
    File image = fs.open(path.c_str(), FILE_WRITE);

    //Increment the image counter (next image)
    image++;
  }

  else {
    //If SD Card is full then exit the application!
    Serial.println("SD CARD FULL ... EXITING!");
    exit();
  }
}

void saveData() {
   output = String(temperature) + String(humidity) + String(image);
   //add the current data to the file
   appendFile(data, "/data.txt", output.toString());
}
```