**Smart Agriculture Monitoring**

|  |  |
| --- | --- |
| **Sr No.** | **Aim of Module** |
| **1.** | **Interfacing Bolt hardware with Arduino**  **THEORY:** Bolt development board can be interfaced with Arduino boards through UART. There are two options to interface in UART: Hardware Serial (Rx and Tx pins) and Software Serial.  **DIAGRAM:**    **PROGRAM CODE:**  #include <BoltDeviceCredentials.h>  #include <BoltIoT-Arduino-Helper.h>  void setup() {  Serial.begin(9600);  boltiot.Begin(Serial); //Hardware UART  }  void loop() {  boltiot.processPushDataCommand(100,25,47,99,"true");  Serial.println(boltiot.getReceivedString());  delay(5000);    //Send data to the Bolt Cloud, when the Bolt polls the Arduino for data.  //This function needs to be called regularly. Calling the CheckPoll function once every seconds is required  } |

|  |  |
| --- | --- |
| **Sr no.** | **Aim of Module** |
| **2.** | **Using Bolt IoT Python Library to control Bolt Hardware**  **THEORY:** Bolt can be controlled through Bolt Cloud using Python. API is available to connect to Cloud. The API has limited functions. Only functions to read and write GPIO data, UART functions and checking device status are present. Commands to push data manually like in Arduino is not possible.  **PROGRAM CODE:**  from boltiot import Bolt  api\_key = "XXXXXXXXXXXXXXXXXXXXXXXX"  device\_id = "BOLTXXXXX"  mybolt = Bolt(api\_key, device\_id)  print(mybolt.isOnline()) |

|  |  |
| --- | --- |
| **Sr No.** | **Aim of Module** |
| **3.** | **Visualizing sensor data using Bolt Cloud**  **THEORY:** Using Bolt Arduino Library, the Gateway Node is now capable to push data to cloud  **PROGRAM CODE:**  **1. For Gateway Node**  // Libraries for Firebase -----------------------------------------------------------------------------------------------------------------------------  #include <FirebaseESP8266.h>  #include <FirebaseESP8266HTTPClient.h>  #include <FirebaseJson.h>  // Libraries to setup Access Point and Station Point ---------------------------------------------------------------------------------------------------  #include <ESP8266WiFi.h>  #include <WiFiClient.h>  #include <ESP8266WiFiMulti.h>  #include <ESP8266WebServer.h>  // Libraries for sending Email -------------------------------------------------------------------------------------------------------------------------  #include <WiFiServer.h>  #include <WiFiServerSecure.h>  #include <WiFiUdp.h>  // Generic Library for Adafruit Sensors-----------------------------------------------------------------------------------------------------------------  #include <Adafruit\_Sensor.h>  // Library for Bolt -----------------------------------------------------------------------------------------------------------------  #include <BoltDeviceCredentials.h>  #include <BoltIoT-Arduino-Helper.h>  // Setting Global Variables ----------------------------------------------------------------------------------------------------------------------------  #define FIREBASE\_HOST "" //Link and Secret Key for Firebase Authentication  #define FIREBASE\_AUTH ""  const char\* APssid = "ESP8266-Access-Point"; //Credentials for Access POint  const char\* APpassword = "123456789";  const char\* ssid = "Khatri"; //Credentials for connecting to a WiFi  const char\* password = "sameep18";  char smtp\_server[] = "mail.smtp2go.com"; // The SMTP Server  String message=""; //Global String variable to store data from sensor nodes  ESP8266WebServer server(80); //Server Object. Port number is 80 for HTTP  WiFiClient espClient; //Client Object to send Emails  FirebaseData firebaseObj; //Firebase Object which will contain path and payload  // Setting up WiFi, WebServer and Firebase -----------------------------------------------------------------------------------------------------  void setup() {  Serial.begin(115200);  boltiot.Begin(12,14);  delay(10);    // Setting up WiFi -------------------------------  Serial.println();  Serial.print("Configuring access point...");    WiFi.mode(WIFI\_AP\_STA);  WiFi.softAP(APssid, APpassword);  WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }    Serial.println("");  Serial.println("WiFi connected");    // Setting up the server ------------------------  server.on("/",handleGenericArgs); //handleGenericArgs is a handler function that will be called whenever '/' is requested by a client  server.on("/sensor",handleSpecificArg); //Similarly handleSpecificArg is also a handler function  server.begin(); //Starting the server  Serial.println("Server started");    // Printing Local and Access Point IP addresses ----------------------------------  Serial.println(WiFi.localIP());  Serial.println(WiFi.softAPIP());    Firebase.begin(FIREBASE\_HOST, FIREBASE\_AUTH); //Initialising Firebase connection  }  void loop() {  // put your main code here, to run repeatedly:  server.handleClient(); //Handles Client Requests  }  //Handler Functions ----------------------------------------------------------------------------------------------------------------------------------  void handleGenericArgs(){  /\*message = "Number of args received: ";  message+=server.args();  message+="\n";\*/  for (int i=0;i<server.args();i++)  {  message += (String)i+"->"; //Collecting Data from Client  message += server.argName(i)+": ";  message += server.arg(i)+"\n";  }  Serial.println(message);  server.send(200,"text/plain","Received Request");  }  void handleSpecificArg()  {  if (!server.hasArg("ldrstatus") && server.arg("ldrstatus")==NULL && !server.hasArg("temp") && server.arg("temp")== NULL  && !server.hasArg("hum") && server.arg("hum")== NULL  && !server.hasArg("soilMoisture") && server.arg("soilMoisture")== NULL) //Sending response to client that data is invalid  {  message="Data Invalid";  server.send(400,"text/plain",message);  }  else  {  String ldr\_status = (String)server.arg("ldrstatus");  String temp = (String)server.arg("temp");  String hum = (String)server.arg("hum");  String soilMoisture = (String)server.arg("soilMoisture");    message="LDR Status = "+ldr\_status;  message+="\nTemperature = "+temp+"C";  message+="\nHumidity = "+hum+"%";  message+="\nSoil Moisture = "+soilMoisture+"%";  Serial.println(message);  byte ret = sendEmail(message); //Sending email whenever handler function is called and data is valid  if (ret)  Serial.println("Email Sent Successully!");  else  Serial.println("Error in sending email!");  Firebase.setString(firebaseObj,"Smart-agriculture/user/1/details/ldr",ldr\_status); //Pushing data to Firebase  Firebase.setInt(firebaseObj,"Smart-agriculture/user/1/details/temp",temp.toInt());  Firebase.setInt(firebaseObj,"Smart-agriculture/user/1/details/hum",hum.toInt());  Firebase.setInt(firebaseObj,"Smart-agriculture/user/1/details/soil",soilMoisture.toInt());  boltiot.processPushDataCommand(ldr\_status == "Light" ? 100 : 0,temp.toInt(),hum.toInt(),soilMoisture.toInt(),"true");  server.send(200,"text/plain","Received Data"); //Sending repsonse to client that valid data is received  }  }  // Function Declaration to Send Email --------------------------------------------------------------------------------------------------------------------  byte sendEmail(String sensor\_data)  {    if (espClient.connect(smtp\_server, 2525) == 1) //Establishing connection to mail server  {  Serial.println(F("connected"));  }  else  {  Serial.println(F("connection failed"));  return 0;  }  if (!emailResp())  return 0;  Serial.println(F("Sending EHLO")); //Sending EHLO to server. It is equivalent to greeting the server. Initially it was HELO but was changed later  espClient.println("EHLO www.example.com"); //Sending www.example.com as a domain. It is a reserved domain that can be used without any permission  if (!emailResp())  return 0;  Serial.println(F("Sending auth login")); //Sending authentication  espClient.println("AUTH LOGIN");  if (!emailResp())  return 0;  Serial.println(F("Sending User")); //Sending Username of SMTP2GO account  espClient.println(""); // Encoded Username in Base64 format. You can calculate it from base64encoded.org  if (!emailResp())  return 0;  Serial.println(F("Sending Password"));  espClient.println(""); //Encoded Password in Base64 Format  if (!emailResp())  return 0;  Serial.println(F("Sending From"));  espClient.println(F("MAIL From: ")); // Enter Sender Mail Id. DON'T FORGET TO CHANGE THE EMAIL ADDRESS  if (!emailResp())  return 0;  Serial.println(F("Sending To"));  espClient.println(F("RCPT To: ")); // Enter Receiver Mail Id. DON'T FORGET TO CHANGE THE EMAIL ADDRESS  if (!emailResp())  return 0;  Serial.println(F("Sending DATA"));  espClient.println(F("DATA"));  if (!emailResp())  return 0;  Serial.println(F("Sending email"));  espClient.println(F("To: ")); // Enter Receiver Mail Id. DON'T FORGET TO CHANGE THE EMAIL ADDRESS  espClient.println(F("From: ")); // Enter Sender Mail Id. DON'T FORGET TO CHANGE THE EMAIL ADDRESS  espClient.println(F("Subject: Sensor Status\r\n"));  espClient.println(sensor\_data);  espClient.println(F("."));  if (!emailResp())  return 0;  Serial.println(F("Sending QUIT")); //Sending QUIT Indicating Email is over  espClient.println(F("QUIT"));  if (!emailResp())  return 0;  espClient.stop();  Serial.println(F("disconnected")); //Disconnecting from Mail Server  return 1;  }    byte emailResp() //Function to handle response by Mail Server  {  byte responseCode;  byte readByte;  int loopCount = 0;    while (!espClient.available())  {  delay(1);  loopCount++;  if (loopCount > 20000)  {  espClient.stop();  Serial.println(F("\r\nTimeout"));  return 0;  }  }  responseCode = espClient.peek();  while (espClient.available())  {  readByte = espClient.read();  Serial.write(readByte);  }  if (responseCode >= '4')  {  return 0;  }  return 1;  }  **2. For Sensor Node**  #include <WiFi.h>  #include <HTTPClient.h>  #include <DHT.h>    #define DHTPIN 34  #define DHTTYPE DHT11  const char\* ssid = "ESP8266-Access-Point";  const char\* password ="123456789";  int ldr=3;  int soilPin=22;  String ldr\_status;  String temp;  String hum;  String soilMoisture;  String payLoad;  DHT dht(DHTPIN,DHTTYPE);  void setup() {    pinMode(ldr,INPUT);  pinMode(soilPin,INPUT);  pinMode(DHTPIN,INPUT);  pinMode(soilPin,INPUT);    Serial.begin(115200);  delay(2000); //Delay needed before calling the WiFi.begin    WiFi.begin(ssid, password);    while (WiFi.status() != WL\_CONNECTED) { //Check for the connection  delay(1000);  Serial.println("Connecting to WiFi..");  }    Serial.println("Connected to the WiFi network : "+(String)ssid);  }  void readLDRStatus(){  if (digitalRead(ldr)==HIGH)  {  ldr\_status="ldrstatus=No Light";  }  else  {  ldr\_status="ldrstatus=Light";  }  }  void readDHTStatus()  {  float t = dht.readTemperature();  float h = dht.readHumidity();  temp="temp="+(String)t;  hum="hum="+(String)h;  }  void readSoilMoisture()  {  int value= analogRead(soilPin);  value = map(value,0,4055,100,0);  soilMoisture="soilMoisture="+(String)value;  }  void generatePayload()  {  readLDRStatus();  readDHTStatus();  readSoilMoisture();  payLoad=ldr\_status+"&"+temp+"&"+hum+"&"+soilMoisture;  Serial.println(payLoad);  }  void loop() {  generatePayload();  if(WiFi.status()== WL\_CONNECTED){ //Check WiFi connection status  HTTPClient http;  http.begin("http://192.168.4.1/light"); //Specify destination for HTTP request    http.addHeader("Content-Type", "application/x-www-form-urlencoded"); //Specify content-type header  int httpResponseCode = http.POST(payLoad); //Send the actual POST request    if(httpResponseCode==200){    //String response = http.getString(); //Get the response to the request    Serial.println(httpResponseCode); //Print return code  //Serial.println(response); //Print request answer    }  else  {  Serial.print("Error on sending POST: ");  Serial.println(httpResponseCode);  }    http.end(); //Free resources    }else{    Serial.println("Error in WiFi connection");    }    delay(10000); //Send a request every 10 seconds  }  **OUTPUT:** |