# Opis razvijenog proizvoda

U okviru ovoga dijela projekta cilj je uspostaviti komunikaciju Arduina i web aplikacija koristeći LoRaWAN. Na Arduino pločicu spojeni su senzori temperature i vlage te infracrveni senzor za detekciju. Rabi se The Things Network aplikacija koja je namijenjena radu s IoT uređajima. U nastavku su dane upute za instalaciju potrebnih alata i konfiguraciju istih za moguću izvedbu spajanja senzora spojenih na Arduino s The Things Network platformom. Također su priloženi kodovi za spajanje pojedinih senzora i spajanje sa platformom.

# Tehničke značajke

Korištena je pločica Arduino Uno i na nju je spojena pločica RFM95W koja predstavlja sklop za LoRaWAN komunikaciju.

# Upute za korištenje

## Instalacija i postavljanje Arduino IDE-a

### Preuzimanje i instalacija Arduino IDE-a

Sa sljedeće stranice preuzeti verziju koja odgovara operacijskom sustavu: <https://www.arduino.cc/en/software>.   
Nakon preuzimanja treba provesti instalaciju po službenim uputama (kliknuti next na sve ponuđeno) i kad je instalacija završena, pokrenuti Arduino IDE iz izbornika Start ili Desktop-a ukoliko je odabrana opcija za Desktop prečicu.

### Konfiguracija library-ja za senzor temperature i vlage

Potrebno je dohvatiti dva library-ja. Prvo skinuti Adafruit\_Sensor library, a zatim DHT library.  
Prvi library moguće je pronaći na linku: <https://github.com/adafruit/Adafruit_Sensor>  
Drugi library nalazi se na linku: <https://github.com/adafruit/DHT-sensor-library>   
Sljedeće upute za konfiguraciju library-ja ponoviti za oba library-ja.  
Library se dohvaća na način da otvorimo link -> kliknemo na Code(sljedeća slika gornji desni kut) -> i zatim download as ZIP (druga oznaka na slici ispod).  
Taj zip library spremimo u neki folder koji želimo (preporuka Documents/Arduino/libraries).

A screenshot of a computer

Description automatically generated with medium confidence

Zatim otvorimo Arduino IDE -> u gornjem izborniku Skica -> Include Library -> Add .ZIP library -> i zatim odaberemo library koji smo dohvatili u koraku prije.

Graphical user interface, text, application

Description automatically generated

### Postavljanje The Things Network-a

Kreirati korisnički račun (*account*) na sljedećoj poveznici: <https://www.thethingsnetwork.org/>  
Nakon stvaranja korisničkog računa prijaviti se na svoj account. Zatim kliknuti na *Start building* na sredini ekrana:

Graphical user interface, text, application

Description automatically generated

Otvara se stranica za odabir servera, gdje odaberemo Europe 1.  
Sada je potrebno kreirati aplikaciju koju će koristiti naš Arduino. U sljedećem izborniku odaberemo *Go to applications* te klikom na *Add application* započnemo kreiranje nove aplikacije. Popunimo potrebna polja za aplikaciju, npr. kao na slici:

Graphical user interface, text, application, email

Description automatically generated

Sada kada je aplikacija kreirana potrebno je dodati *End device* u tu aplikaciju. *End device* zapravo predstavlja naš konkretni Arduino-LoRaWAN spoj koji će slati podatke na *gateway*. Na otvorenom ekranu odaberemo opciju *Add end device*.



Otvara se sljedeći prozor na kojem je potrebno odabrati opciju *Manually:*

Graphical user interface, text, application, email

Description automatically generated

Postavimo parametre kao na sljedeće dvije slike (opcije koje imaju *Generate* pokraj sebe odabrati to):

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

*Frequency plan* predstavlja način rada komunikacije (frekvenciju i način kodiranja). LoRaWAN *version* je verzija LoRaWAN-a i možemo postaviti zadnju verziju. *Regional Parameters Version* postavimo na PHY V1.1 REV A.  
Također potrebno je kliknuti *show advanced activation.*Tu odaberemo *OTAA, class A only* jer koristimo uređaj bez baterije.  
Sljedeća tri polja generiramo koristeći ugrađene funkcije, a Join EUI upišemo onaj koji ćemo upisati i u kodu za spajanje.

### Kod za Arduino i spoj

#### Spoj infracrvenog senzora za detekciju

Graphical user interface, application

Description automatically generated

#### Kod za rad infracrvenog senzora za detekciju

1. #define IRPIN 2 // Digital pin connected to the IR sensor
2. // Connect pin 1 (on the left) of the sensor to your defined IRPIN
3. // Connect pin 2 of the sensor to GROUND
4. // Connect pin 3 (on the right) of the sensor to +5V
5. void setup() {
6. pinMode (IRPIN, INPUT); // sensor pin INPUT
7. Serial.begin(9600);
8. Serial.println("IR readings:");
9. }
10. void loop() {
11. delay(1000);
12. // Read IR status
13. int statusSensor = digitalRead (IRPIN);
14. if (statusSensor == 1) {
15. Serial.println("No person detected.\n");
16. }
17. else {
18. Serial.println("PERSON DETECTED!\n");
19. }
20. }

#### Spoj senzora temperature i vlage

A picture containing text, electronics, circuit

Description automatically generated

#### Kod za rad senzora temperature i vlage

1. #include "DHT.h"
2. #define DHTPIN 8 // Digital pin connected to the DHT sensor
3. #define DHTTYPE DHT11
4. // Connect pin 1 (on the left) of the sensor to +5V
5. // Connect pin 2 of the sensor to your defined DHTPIN
6. // Connect pin 3 (on the right) of the sensor to GROUND
7. // Initialize DHT sensor.
8. DHT dht(DHTPIN, DHTTYPE);
9. void setup() {
10. Serial.begin(9600);
11. Serial.println(F("DHTxx readings:"));
12. dht.begin();
13. }
14. void loop() {
15. delay(1000);
16. // Reading temperature or humidity takes about 250 milliseconds!
17. float h = dht.readHumidity();
18. // Read temperature as Celsius (the default)
19. float t = dht.readTemperature();
20. // Read temperature as Fahrenheit (isFahrenheit = true)
21. float f = dht.readTemperature(true);
22. // Check if any reads failed and exit early (to try again).
23. if (isnan(h) || isnan(t) || isnan(f)) {
24. Serial.println(F("Failed to read from DHT sensor!"));
25. return;
26. }
27. // Compute heat index in Fahrenheit (the default)
28. float hif = dht.computeHeatIndex(f, h);
29. // Compute heat index in Celsius (isFahreheit = false)
30. float hic = dht.computeHeatIndex(t, h, false);
31. Serial.print(F("Humidity: "));
32. Serial.print(h);
33. Serial.print(F("% Temperature: "));
34. Serial.print(t);
35. Serial.print(F("°C "));
36. Serial.print(f);
37. Serial.print(F("°F Heat index: "));
38. Serial.print(hic);
39. Serial.print(F("°C "));
40. Serial.print(hif);
41. Serial.println(F("°F"));
42. }

#### Kod za dobivanje informacija o uređaju potrebnih za registraciju uređaja

1. #include <TheThingsNetwork.h>
2. #include <SoftwareSerial.h>
3. #define debugSerial Serial
4. SoftwareSerial loraSerial(10, 11); // RX, TX
5. #define freqPlan TTN\_FP\_EU868
6. TheThingsNetwork ttn(loraSerial, debugSerial, freqPlan);
7. void setup()
8. {
9. loraSerial.begin(9600);
10. debugSerial.begin(9600);
11. }
12. void loop()
13. {
14. debugSerial.println("Device Information");
15. debugSerial.println();
16. ttn.showStatus();
17. debugSerial.println();
18. debugSerial.println("Use the EUI to register the device for OTAA");
19. debugSerial.println("-------------------------------------------");
20. debugSerial.println();
21. delay(1000);
22. }

#### Kod za slanje Hello world na The Things Network

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. \* Copyright (c) 2015 Thomas Telkamp and Matthijs Kooijman
3. \* Copyright (c) 2018 Terry Moore, MCCI
4. \*
5. \* Permission is hereby granted, free of charge, to anyone
6. \* obtaining a copy of this document and accompanying files,
7. \* to do whatever they want with them without any restriction,
8. \* including, but not limited to, copying, modification and redistribution.
9. \* NO WARRANTY OF ANY KIND IS PROVIDED.
10. \*
11. \* This example sends a valid LoRaWAN packet with payload "Hello,
12. \* world!", using frequency and encryption settings matching those of
13. \* the The Things Network.
14. \*
15. \* This uses OTAA (Over-the-air activation), where where a DevEUI and
16. \* application key is configured, which are used in an over-the-air
17. \* activation procedure where a DevAddr and session keys are
18. \* assigned/generated for use with all further communication.
19. \*
20. \* Note: LoRaWAN per sub-band duty-cycle limitation is enforced (1% in
21. \* g1, 0.1% in g2), but not the TTN fair usage policy (which is probably
22. \* violated by this sketch when left running for longer)!
23. \* To use this sketch, first register your application and device with
24. \* the things network, to set or generate an AppEUI, DevEUI and AppKey.
25. \* Multiple devices can use the same AppEUI, but each device has its own
26. \* DevEUI and AppKey.
27. \*
28. \* Do not forget to define the radio type correctly in
29. \* arduino-lmic/project\_config/lmic\_project\_config.h or from your BOARDS.txt.
30. \*
31. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
32. #include <lmic.h>
33. #include <hal/hal.h>
34. #include <SPI.h>
35. //
36. // For normal use, we require that you edit the sketch to replace FILLMEIN
37. // with values assigned by the TTN console. However, for regression tests,
38. // we want to be able to compile these scripts. The regression tests define
39. // COMPILE\_REGRESSION\_TEST, and in that case we define FILLMEIN to a non-
40. // working but innocuous value.
41. //
42. #ifdef COMPILE\_REGRESSION\_TEST
43. # define FILLMEIN 0
44. #else
45. # warning "You must replace the values marked FILLMEIN with real values from the TTN control panel!"
46. # define FILLMEIN (#dont edit this, edit the lines that use FILLMEIN)
47. #endif
48. // This EUI must be in little-endian format, so least-significant-byte
49. // first. When copying an EUI from ttnctl output, this means to reverse
50. // the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,
51. // 0x70.
52. static const u1\_t PROGMEM APPEUI[8]={ 0x70, 0xB3, 0xD5, 0x7E, 0xD0, 0x03, 0xB1, 0x8C };
53. void os\_getArtEui (u1\_t\* buf) { memcpy\_P(buf, APPEUI, 8);}
54. // This should also be in little endian format, see above.
55. static const u1\_t PROGMEM DEVEUI[8]={ 0x62, 0x9B, 0x04, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 };
56. void os\_getDevEui (u1\_t\* buf) { memcpy\_P(buf, DEVEUI, 8);}
57. // This key should be in big endian format (or, since it is not really a
58. // number but a block of memory, endianness does not really apply). In
59. // practice, a key taken from ttnctl can be copied as-is.
60. static const u1\_t PROGMEM APPKEY[16] = { 0xC8, 0xE8, 0x65, 0x33, 0xE2, 0x19, 0x9C, 0x09, 0xA8, 0x70, 0xD8, 0x8D, 0x70, 0xF1, 0x86, 0x7E };
61. void os\_getDevKey (u1\_t\* buf) { memcpy\_P(buf, APPKEY, 16);}
62. static uint8\_t mydata[] = "Hello, world!";
63. static osjob\_t sendjob;
64. // Schedule TX every this many seconds (might become longer due to duty
65. // cycle limitations).
66. const unsigned TX\_INTERVAL = 5;
67. // Pin mapping
68. const lmic\_pinmap lmic\_pins = {
69. .nss = 10,
70. .rxtx = LMIC\_UNUSED\_PIN,
71. .rst = 5,
72. .dio = {2, 3, LMIC\_UNUSED\_PIN},
73. };
74. void printHex2(unsigned v) {
75. v &= 0xff;
76. if (v < 16)
77. Serial.print('0');
78. Serial.print(v, HEX);
79. }
80. void onEvent (ev\_t ev) {
81. Serial.print(os\_getTime());
82. Serial.print(": ");
83. switch(ev) {
84. case EV\_SCAN\_TIMEOUT:
85. Serial.println(F("EV\_SCAN\_TIMEOUT"));
86. break;
87. case EV\_BEACON\_FOUND:
88. Serial.println(F("EV\_BEACON\_FOUND"));
89. break;
90. case EV\_BEACON\_MISSED:
91. Serial.println(F("EV\_BEACON\_MISSED"));
92. break;
93. case EV\_BEACON\_TRACKED:
94. Serial.println(F("EV\_BEACON\_TRACKED"));
95. break;
96. case EV\_JOINING:
97. Serial.println(F("EV\_JOINING"));
98. break;
99. case EV\_JOINED:
100. Serial.println(F("EV\_JOINED"));
101. {
102. u4\_t netid = 0;
103. devaddr\_t devaddr = 0;
104. u1\_t nwkKey[16];
105. u1\_t artKey[16];
106. LMIC\_getSessionKeys(&netid, &devaddr, nwkKey, artKey);
107. Serial.print("netid: ");
108. Serial.println(netid, DEC);
109. Serial.print("devaddr: ");
110. Serial.println(devaddr, HEX);
111. Serial.print("AppSKey: ");
112. for (size\_t i=0; i<sizeof(artKey); ++i) {
113. if (i != 0)
114. Serial.print("-");
115. printHex2(artKey[i]);
116. }
117. Serial.println("");
118. Serial.print("NwkSKey: ");
119. for (size\_t i=0; i<sizeof(nwkKey); ++i) {
120. if (i != 0)
121. Serial.print("-");
122. printHex2(nwkKey[i]);
123. }
124. Serial.println();
125. }
126. // Disable link check validation (automatically enabled
127. // during join, but because slow data rates change max TX
128. // size, we don't use it in this example.
129. LMIC\_setLinkCheckMode(0);
130. break;
131. /\*
132. || This event is defined but not used in the code. No
133. || point in wasting codespace on it.
134. ||
135. || case EV\_RFU1:
136. || Serial.println(F("EV\_RFU1"));
137. || break;
138. \*/
139. case EV\_JOIN\_FAILED:
140. Serial.println(F("EV\_JOIN\_FAILED"));
141. break;
142. case EV\_REJOIN\_FAILED:
143. Serial.println(F("EV\_REJOIN\_FAILED"));
144. break;
145. case EV\_TXCOMPLETE:
146. Serial.println(F("EV\_TXCOMPLETE (includes waiting for RX windows)"));
147. if (LMIC.txrxFlags & TXRX\_ACK)
148. Serial.println(F("Received ack"));
149. if (LMIC.dataLen) {
150. Serial.print(F("Received "));
151. Serial.print(LMIC.dataLen);
152. Serial.println(F(" bytes of payload"));
153. }
154. // Schedule next transmission
155. os\_setTimedCallback(&sendjob, os\_getTime()+sec2osticks(TX\_INTERVAL), do\_send);
156. break;
157. case EV\_LOST\_TSYNC:
158. Serial.println(F("EV\_LOST\_TSYNC"));
159. break;
160. case EV\_RESET:
161. Serial.println(F("EV\_RESET"));
162. break;
163. case EV\_RXCOMPLETE:
164. // data received in ping slot
165. Serial.println(F("EV\_RXCOMPLETE"));
166. break;
167. case EV\_LINK\_DEAD:
168. Serial.println(F("EV\_LINK\_DEAD"));
169. break;
170. case EV\_LINK\_ALIVE:
171. Serial.println(F("EV\_LINK\_ALIVE"));
172. break;
173. /\*
174. || This event is defined but not used in the code. No
175. || point in wasting codespace on it.
176. ||
177. || case EV\_SCAN\_FOUND:
178. || Serial.println(F("EV\_SCAN\_FOUND"));
179. || break;
180. \*/
181. case EV\_TXSTART:
182. Serial.println(F("EV\_TXSTART"));
183. break;
184. case EV\_TXCANCELED:
185. Serial.println(F("EV\_TXCANCELED"));
186. break;
187. case EV\_RXSTART:
188. /\* do not print anything -- it wrecks timing \*/
189. break;
190. case EV\_JOIN\_TXCOMPLETE:
191. Serial.println(F("EV\_JOIN\_TXCOMPLETE: no JoinAccept"));
192. break;
193. default:
194. Serial.print(F("Unknown event: "));
195. Serial.println((unsigned) ev);
196. break;
197. }
198. }
199. void do\_send(osjob\_t\* j){
200. // Check if there is not a current TX/RX job running
201. if (LMIC.opmode & OP\_TXRXPEND) {
202. Serial.println(F("OP\_TXRXPEND, not sending"));
203. } else {
204. // Prepare upstream data transmission at the next possible time.
205. LMIC\_setTxData2(1, mydata, sizeof(mydata)-1, 0);
206. Serial.println(F("Packet queued"));
207. }
208. // Next TX is scheduled after TX\_COMPLETE event.
209. }
210. void setup() {
211. Serial.begin(9600);
212. Serial.println(F("Starting"));
213. #ifdef VCC\_ENABLE
214. // For Pinoccio Scout boards
215. pinMode(VCC\_ENABLE, OUTPUT);
216. digitalWrite(VCC\_ENABLE, HIGH);
217. delay(1000);
218. #endif
219. // LMIC init
220. os\_init();
221. // Reset the MAC state. Session and pending data transfers will be discarded.
222. LMIC\_reset();
223. // Start job (sending automatically starts OTAA too)
224. do\_send(&sendjob);
225. }
226. void loop() {
227. os\_runloop\_once();
228. }

# Literatura

[1] Matijević E., Implementacija LoRaWAN tehnologije u okruženju pametnoga doma, Diplomski rad, Fakultet elektrotehnike i računarstva, 2021.

[2] Službena dokumentacija The Things Network-a, https://www.thethingsnetwork.org/docs/