IoT Leaf

Preliminary Users Manual

Table of Contents

1 Copyright

2 History

Date	Changes in document
06/2019	Functional blocks description added
06/2019	Description of battery connector extended
05/2019	Initial version of preliminary manual

3 Safety

The hardware described here is an electrostatic sensitive device. This means it can be damaged by common static charges which build up on people, tools and other non-conductors or semiconductors. To avoid such a damage, it has to be handled with care and including all relevant procedures (like proper grounding of people handling the devices, shielding/covering to not to let a person touch the device unwanted, proper packaging in ESD-bags, ...). For more information please refer to related regulations and standards regarding handling of ESD devices.

The hardware described here is a component which is intended to be used as part of a larger device.

This document describes the IoT Leaf-hardware but may contain errors or may be changed without further notice.

4 Overview

This document describes the IoT Leaf embedded module, its electrical characteristics and usage. It is designed for various IoT applications and can be used in different environments and scenarios – even in self-contained, solar-powered devices. It comes with full hardware capabilities and with an example firmware that demonstrates some of its features. It is not a ready-to-use device but a component which is intended to be integrated in larger devices and which has to be equipped with an own firmware controlling these devices. Development and integration of this firmware is up to the user.

4.1 Features

The IoT Leaf consist of two boards which are connected to each other and can be separated by breaking them off along the predefined line. One board contains the power supply with support for USB-power, solar cells, LiPo battery and the other one contains the SoC with the peripheral interfaces. It can be used in following ways:

- both together can be used as one complete, ready-to-use unit
- both separate (broken out of each other along the drill holes) can be used as a power supply for any other 3.3V device and as a microcontroller board for any suitable application
- both reconnected (broken out of each other along the drill holes and then soldered together via the 3.3V voltage output and input pads using an angular pin header like the multicomp MC34745 angular header; please ensure proper isolation of the pins to not to cause a short-circuit) used as one complete, ready-to-use unit;
  in this variant please note that the signals from the antenna may be partially shielded by the edgeways mounted part of the PCB

Following the features of both are described.

4.1.1 IoT Leaf power supply part

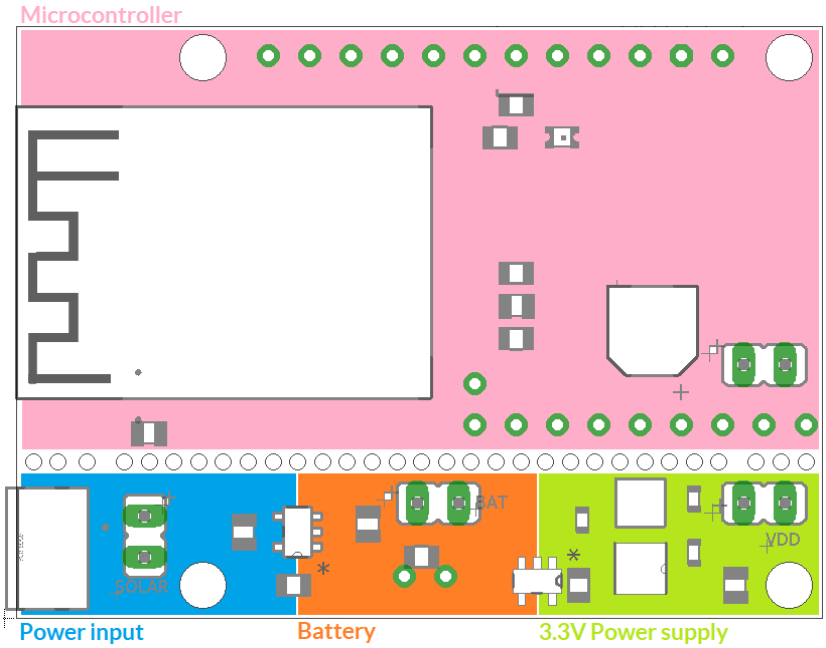
- Power input via micro-USB or via solar panel for green and self-contained IoT applications
- Supports LiPo battery cell, automatic charging and deep discharge protection
- Battery charging with up to 450 mA
- Provides 3.3V output out of battery and/or power input with a current of up to 500 mA
- Optional: hardware switch to turn off power and to stay in a lossless battery-charging-mode

4.1.2 IoT Leaf microcontroller part

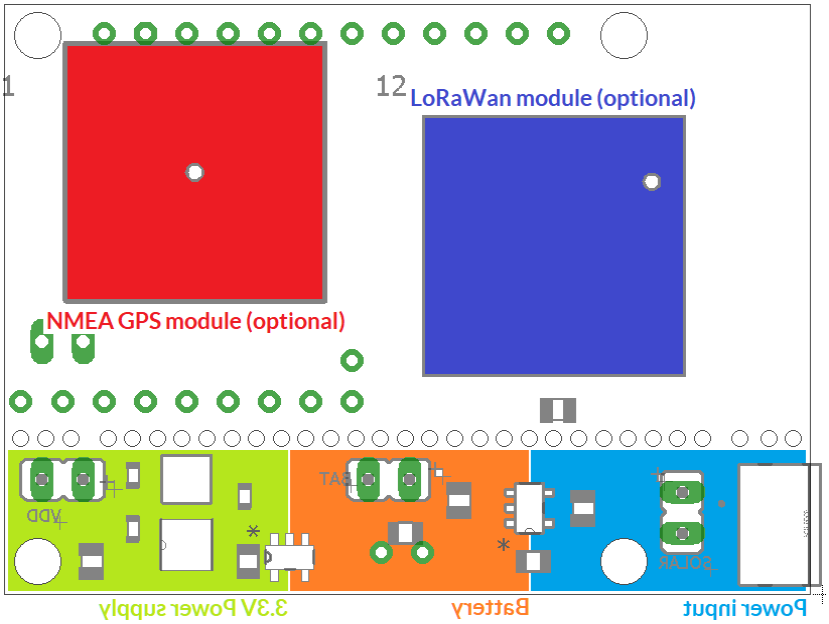
- Bases on Espressif ESP32 dual-core SoC with 16 MByte flash memory, 520 kBytes SRAM and up to 240 MHz CPU-clock (ESP-WROOM-32)
- Supports WiFi in client and access-point mode with on-board antenna
- Provides Bluetooth and Bluetooth LE (BLE, Bluetooth low energy) with included antenna
- Optional: supports LoRaWAN (Long Range Wide Area Network) via RFM95/96/97/98W module
- Optional: supports GPS via L80-M39 module
- Integrated Hall-sensor and different analogue and digital interfaces
- On-board low-power LED for basic signalling tasks
- Can be operated with IoT Leaf power supply board or separated with any other, stabilised 3.3V power source
- Easily programmable via 3.3V serial interface out of ArduinoIDE
- Interfaces, outputs and layout can be modified according to your requirements

5 Boards And Connectors

5.1 Functional blocks

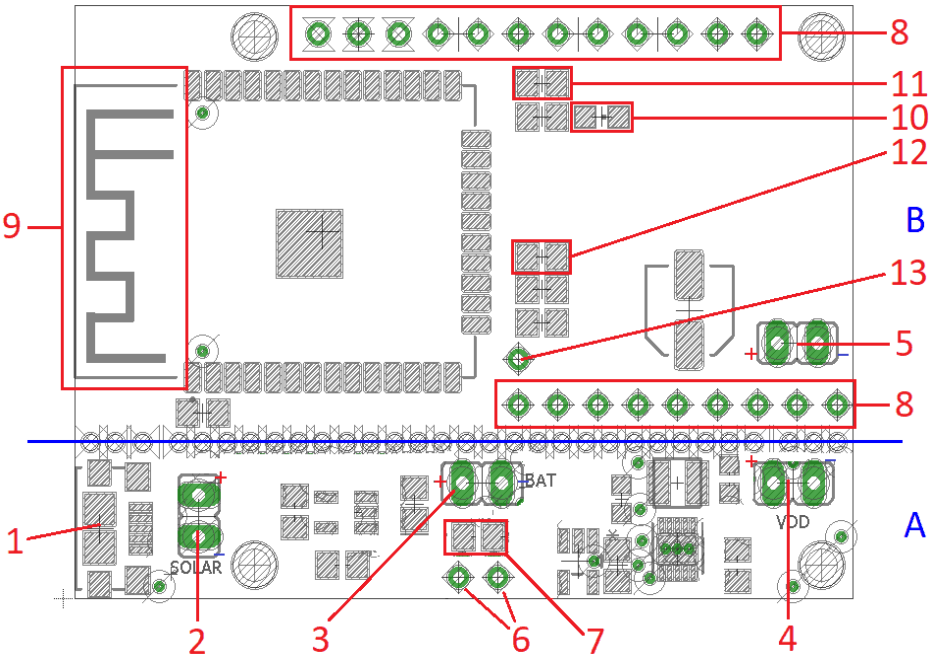


Top view of functional blocks:
turquoise – power input part where power can be provided either via microUSB input or via solar cell; here only one of both, solar or USB power input should be used at the same time
orange – battery part with charge controller, connector for LiPo battery cell, deep-discharge-protection and connector for optional switch to turn off the microcontroller completely while battery charging function is still active
green – power supply part providing stabilised 3.3V to the microcontroller
pink – microcontroller part with signal connectors (for a description of the available signals please refer to "5.3 ESP32 signal connectors" below)



Bottom view of functional blocks:
red – optional NMEA GPS module, when it is used, it is recommended to operate the board botto-up in order to ensure proper receiving of GPS signals
blue – optional LoRaWAN module for long range low power data communication

5.2 General layout



The IoT Leaf board consists of the two parts power supply (A) and microcontroller (B). Both can be operated as one board or they can broken into two separately usable parts along the blue line. The both boards provide following connectors and interfaces:

1. MicroUSB power supply – can be used to power the board and to charge the optional LiPo battery via USB. When this connector is used, under absolutely no conditions power has to be fed into Solar-connector (2), this would lead to an irreversible damage of the board and/or the connected USB device! Depending on the capacity of the optional battery (3) and the current consumption of the whole board it has to be ensured the USB-connector can provide enough current here.

2. Solar power input to be used with a solar cell or with any other power supply. Here DC with a voltage in range 3,75V .. 6V can be provided.
   Please note: A total maximum of 6.5 V in no case has to be exceeded. So when using solar cells, please also check their no-load voltage regarding this limit!
   Please note: When this input is used for powering the board, the microUSB connector (1) in no case has to be used, this would lead to an irreversible damage of the board and/or the connected USB-device!
   On this connector positive pole (+) is the upper one, negative (-) the lower.

3. Optional: battery connection for a single 3.7V LiPo battery cell. When a battery is connected, it is charged with a maximum current of about 450 mA and up to a maximum charging voltage of 4.2 V. During operation and discharging the voltage of the battery is checked, the following hardware is turned off when dropping below a voltage of 3.0 V. Other maximum charging voltages and other deep-discharge-thresholds for other kinds of batteries are available on request.
   Please note: also when the main hardware is turned off, there is still a minimal current of about 50µA still pulled out of the battery. So in case of dropping below the lower limit, ensure to provide power soon in order to avoid a deep discharge and damage of the battery! Alternatively the board also can be turned off via the switch (6) in order to cut off the load completely.
   Please note: choose a suitable LiPo battery with a capacity that fits to the maximum charge current mentioned above in order to avoid damage from the battery!
   Please note: when no battery is used, here a capacitor of about 100 µF has to be placed. In case of an electrolytic capacitor please note the polarity, placing it wrong can damage the board!
   On this connector positive pole (+) is the left one, negative (-) the right.

4. Optional: 3.3V voltage output. This output can be used when the power supply part (A) is used separately. Elsewhere the 3.3V from this board are routed to the microcontroller-part (B) automatically and this connector can be ignored.

On this connector positive pole (+) is the left one, negative (-) the right.

5. Optional: 3.3V voltage input. This input can be used when the microcontroller-part (B) is used separately. Elsewhere the 3.3V from the power supply part (A) to this board are routed automatically and this connector can be ignored.
   On this connector positive pole (+) is the left one, negative (-) the right.

6. Optional: solder pads for a two-pole header which can be used to connect a switch to cut off solar input and battery charging part. Using such a switch the whole board including the microprocessor can be turned off leaving only the solar and battery-charging function active. This ensures the battery is still charged also when the connected control electronic is not used. Before connecting such a switch that, the solder jumper (7) has to be removed, elsewhere the switching function is not guaranteed as this solder jumper bypasses these two solder pads and therefore the function of the switch.

7. Solder jumper to permanently power the connected electronics from power input and/or battery. When a switch via connection at (6) is used, this solder jumper has to be removed/opened.

8. Signal connectors of the microcontroller. These pins provide different signals including 3.3V, GND and RX/TX for programmming the board. For details and an exact pinout please refer section “5.3 ESP32 signal connectors” below.

9. Integrated Bluetooth and WiFi antenna.
   Please keep clear this area from any metals and conductive materials in order to ensure proper operation!

10. Green state LED, can be switched by the firmware. This LED is connected to GPIO4 of the ESP32 module and is a low-power type so that turning on the LED consumes about 4 mA only.

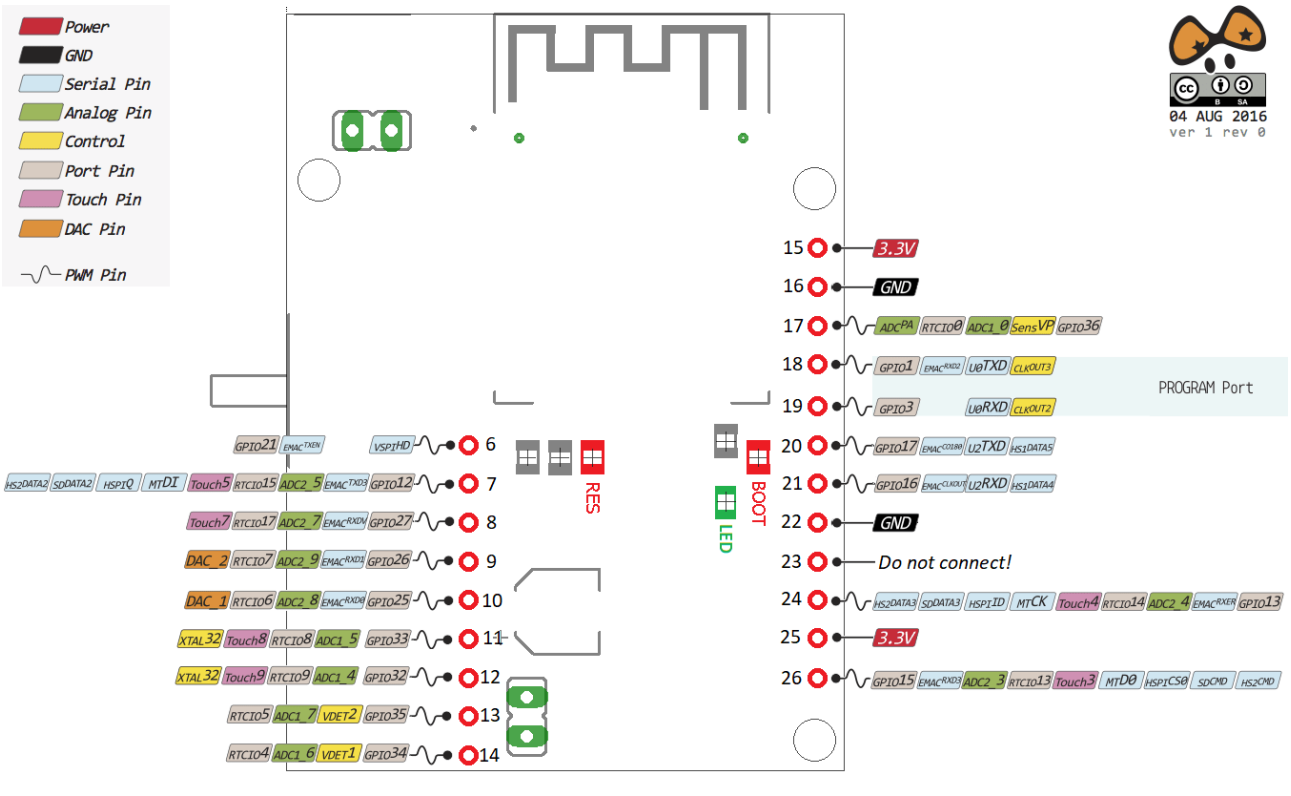
11. Boot-pads, to switch the boot-mode via IO0 of the ESP32 module, these two contacts can be bridged in order to set the boot-state. The board can be programmed via the 3.3V serial interface at pins 18 and 19, to enable writing it may be necessary to short-circuit this boot-jumper to apply GND to GPIO0.

12. Reset-pads, to reboot the controller (turn off/on EN) these two contacts need to be bridged. The ESP32-SoCs do not provide a real reset-mechanism, nevertheless by applying a short-circuit to this solder-jumper, the EN-input is pulled to VDD disabling the SoC

13. Solder point for LoRaWAN antenna (when equipped with optional module, please refer to section “6.1.1 On-Board LoRaWAN Sender / Receiver” for details)

5.3 ESP32 signal connectors

Following signals can be found at the connectors of the microcontroller-part of the IoT Leaf. The signal output pads (marked in red in image below) all come with a 2,54 mm distance and therefore can be used with standard board-to-board connectors or with a breadboard.



Please note:

- All signals come with 3.3V level, applying 5V or any other voltage greater than 3.3V to one of the pins may destroy the board irreversibly
- pins 18 and 19 provide a 3.3V TX and RX serial interface to program the ESP32, they can be used e.g. with any suitable USB-to-serial adapter providing 3.3V signals (such as the Olimex PL2303 3 Pin Cable or the FTDI TTL-232R-RPi cable)
- pin 22 is reserved for future use and in no case should be connected to anything

6 Software

The IoT Leaf bases on the well-documented ESP-WROOM-32 module by Espressif. It can be programmed via the 3.3V serial interface (as described in 5.3 ESP32 signal connectors). Programming is possible e.g. via the ArduinoIDE or via any other development environment that is able to compile properly for this hardware.

When the ArduinoIDE is used, following steps are necessary to write a new firmware to the ESP32 module on the IoT Leaf board:

1. connect the board via a 3.3V serial adapter with the PC using pins 16, 18 and 19 (please refer to "5.3 ESP32 signal connectors")
2. close (short-circuit) the two jumpers BOOT and RES (please refer to "5.3 ESP32 signal connectors")
3. release the connection of the RES-jumper → now the board boots in UART-programming mode which gives the possibility to download a firmware via serial interface
4. release the BOOT-jumper
5. assumed the ArduinoIDE is configured properly to use an ESP32 board: download the new firmware to the controller
6. once the download has completed: perform a full power cycle or trigger the RES-jumper again in order to restart the board
7. now the new firmware is started and can be used

Details about the ESP-WROOM-32 module which is the core of the IoT Leaf board are available at https://www.espressif.com/en/products/hardware/esp-wroom-32/overview.

The ArduinoIDE is available for free at https://www.arduino.cc/en/Main/Software.

A webboard for questions and answers about the ESP32 platform can be found at https://www.esp32.com.

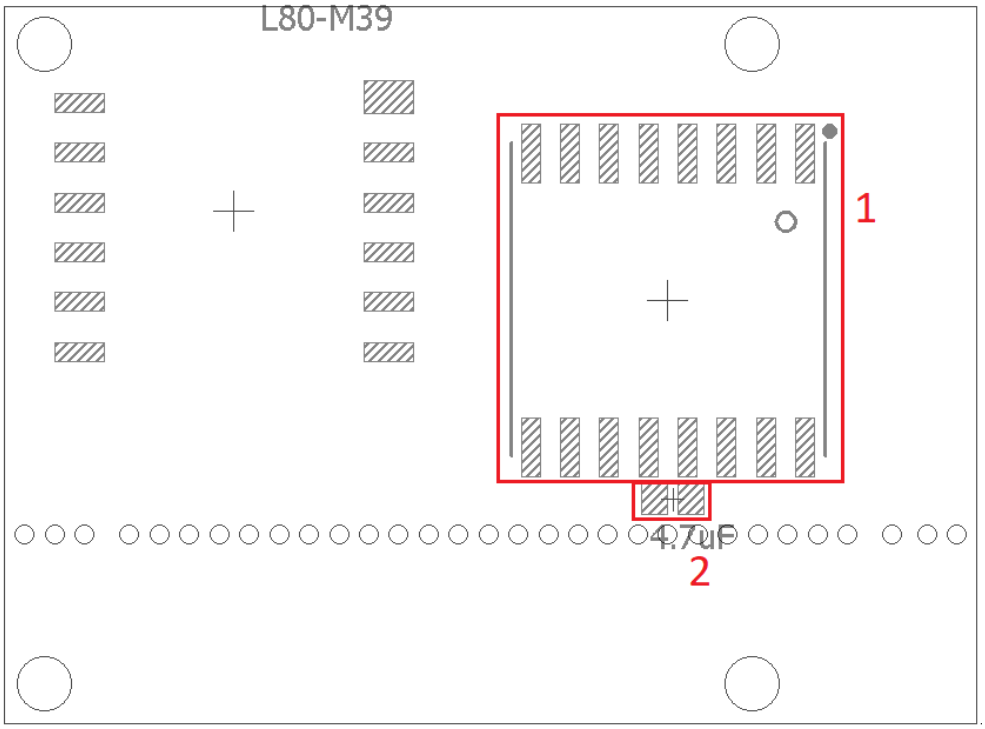
Beside of that there are a lot of other manuals, tutorials and example codes available out there for ESP32 boards / ESP-WROOM-32 modules.

6.1 Application Examples

Following some examples are given showing how to implement features and functions for the IoT Leaf board. This includes some of the optional functions that require additional hardware on the back side of the controller board.

6.1.1 On-Board LoRaWAN Sender / Receiver

To add support for LoRaWAN a RFM95W module (1) as well as a 4.7 µF ceramic capacitor (2) is required on bottom side of the board:



When the module is added manually, beside the capacitor at the module at least the following pins have to be connected: 1, 2, 3, 4, 5, 6, 8, 9, 10, 13 and 14. Nevertheless for a proper mounting and mechanical strength it is recommended to solder all pins.

This module can be accessed via the following hardware signals:

- GPIO2, GPIO5, GPIO18, GPIO19 and GPIO23 for control and communication via SPI interface
- GPIO14 to reset the LoRaWAN module
  Please note: when operating in transmit mode, the module may consume more than 100 mA. This may overload the power supply of the control board and/or quickly empty the connected battery. Thus it is recommended to turn on transmission mode only for a very short time and to take additional measures to save power during this time (e.g. turn off WiFi and/or turn off Bluetooth and/or hold GPS module in reset and/or work with lower ESP32 clock frequency)

A LoRa sender sketch for the ArduinoIDE may look like this (taken from https://randomnerdtutorials.com/esp32-lora-rfm95-transceiver-arduino-ide):

```
#include <SPI.h>
#include <LoRa.h>

//define the pins used by the transceiver module
#define ss 5
#define rst 14
#define dio0 2

int counter = 0;

void setup() {
  //initialize Serial Monitor
  Serial.begin(115200);
  while (!Serial);
  Serial.println("LoRa Sender");

  //setup LoRa transceiver module
  LoRa.setPins(ss, rst, dio0);

  //replace the LoRa.begin(---E-) argument with your location's frequency
  //433E6 for Asia
```

```
  //866E6 for Europe
  //915E6 for North America
  while (!LoRa.begin(866E6)) {
    Serial.println(".");
    delay(500);
  }
  // Change sync word (0xF3) to match the receiver
  // The sync word assures you don't get LoRa messages from other LoRa
  // transceivers
  // ranges from 0-0xFF
  LoRa.setSyncWord(0xF3);
  Serial.println("LoRa Initializing OK!");
}

void loop() {
  Serial.print("Sending packet: ");
  Serial.println(counter);

  //Send LoRa packet to receiver
  LoRa.beginPacket();
  LoRa.print("hello ");
  LoRa.print(counter);
  LoRa.endPacket();

  counter++;

  delay(10000);
}
```

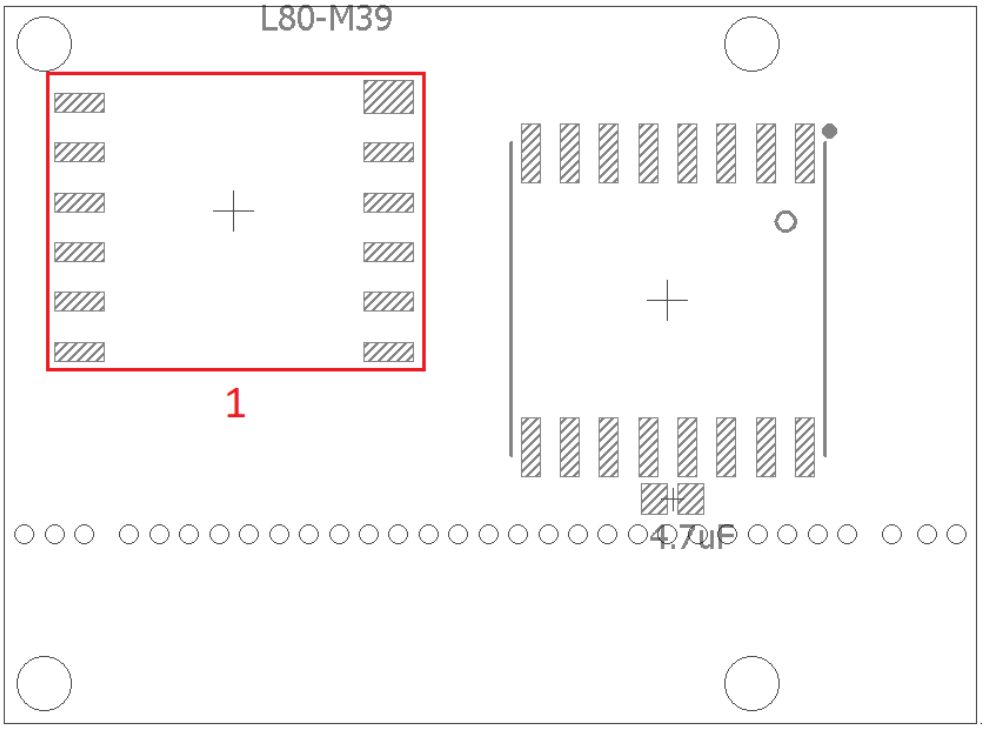
A LoRa receiver sketch for the ArduinoIDE may look like this (taken from https://randomnerdtutorials.com/esp32-lora-rfm95-transceiver-arduino-ide):

```
#include <SPI.h>
#include <LoRa.h>

//define the pins used by the transceiver module
#define ss 5
#define rst 14
#define dio0 2

void setup() {
  //initialize Serial Monitor
  Serial.begin(115200);
  while (!Serial);
  Serial.println("LoRa Receiver");

  //setup LoRa transceiver module
  LoRa.setPins(ss, rst, dio0);

  //replace the LoRa.begin(---E-) argument with your location's frequency
  //433E6 for Asia
  //866E6 for Europe
  //915E6 for North America
  while (!LoRa.begin(866E6)) {
    Serial.println(".");
    delay(500);
  }
  // Change sync word (0xF3) to match the receiver
  // The sync word assures you don't get LoRa messages from other LoRa
  // transceivers
  // ranges from 0-0xFF
  LoRa.setSyncWord(0xF3);
  Serial.println("LoRa Initializing OK!");
```

```
}

void loop() {
  // try to parse packet
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packet
    Serial.print("Received packet '");

    // read packet
    while (LoRa.available()) {
      String LoRaData = LoRa.readString();
      Serial.print(LoRaData);
    }

    // print RSSI of packet
    Serial.print("' with RSSI ");
    Serial.println(LoRa.packetRssi());
  }
}
```

6.1.2 On-Board GPS

To add support for NMEA GPS, a L80-M39 module (1) is required on the bottom side of the board:



When the module is added manually, at least the following pins have to be connected: 1, 2, 3, 4, 5, 7, 10 and 12. Nevertheless for a proper mounting and mechanical strength it is recommended to solder all pins.

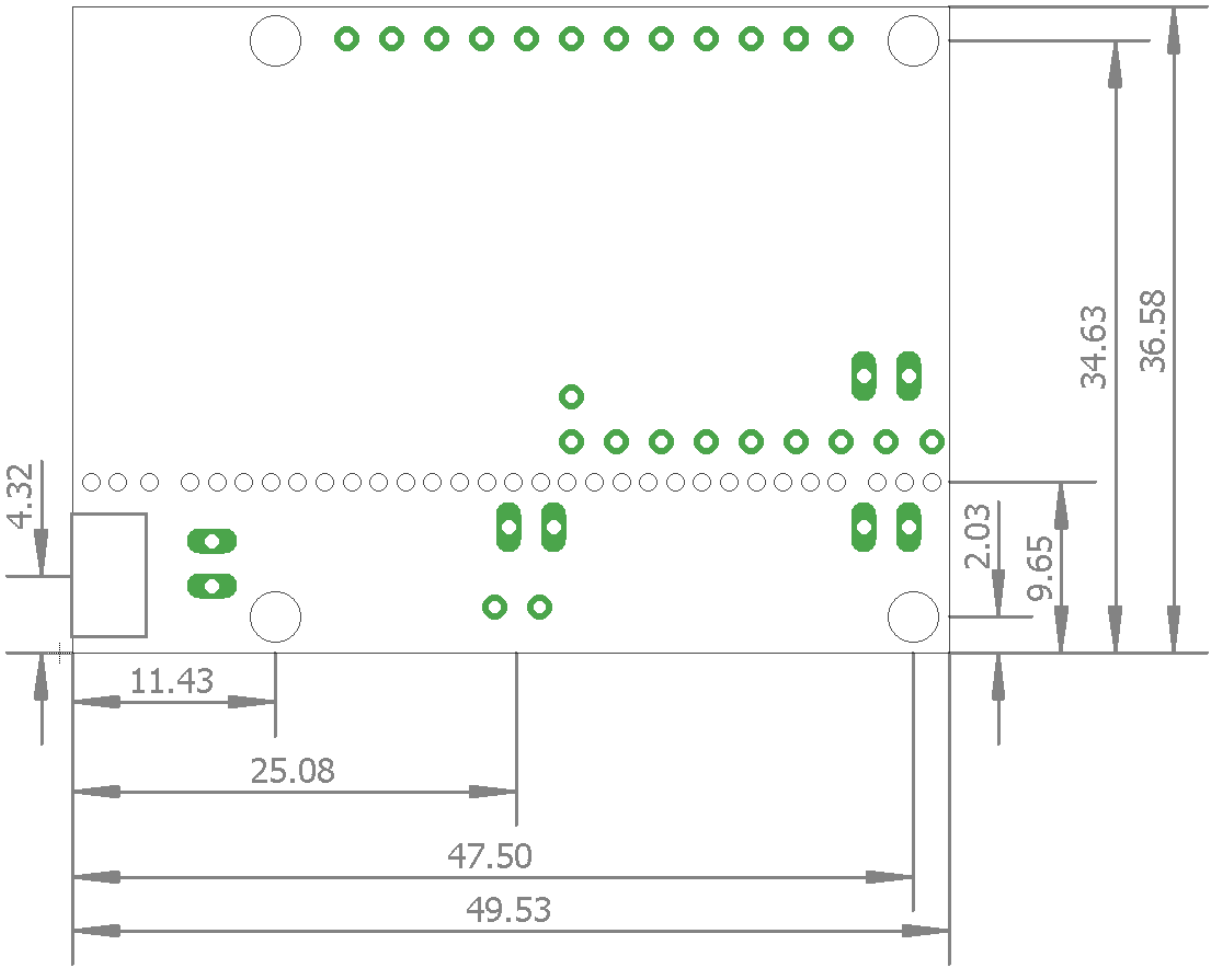
This module can be accessed via the following hardware signals:

- RXD2, TXD2 for serial communication
  Please note: when this module is used, the output pins 20 and 21 (see "5.3 ESP32 signal connectors") can't be used any more as they are occupied for direct communication between ESP32 and GPS module
- GPIO22 to reset the GPS module
  Please note: when operating in transmit mode, the module may consume more than 100 mA. This may overload the power supply of the control board and/or quickly empty the connected battery. Thus it is recommended to turn on transmission mode only for a very short time and to take additional measures to save power during this time (e.g. turn off WiFi and/or turn off Bluetooth and/or hold LoRaWAN module in reset and/or work with lower ESP32 clock frequency)

As soon as the module is taken out of reset via GPIO22 and as soon as there is a GPS fix, the related position information can be retrieved via the serial interface 2 in NMEA format.

APPENDIX A – Board dimensions

Board dimension drawings, all values are given in unit mm.



Alphabetical Index