Learning IoT

Goody for Building IoT in Munich 25. - 27. April 2023.

Images PCB

Pin List

GPIO	Pin	Function
GPIO0	D3	Button / SW1
GPIO2	D4	LED Green
GPIO15	D8	LED Blue
GPIO16	D0	LED Red

Bill of Materials (BOM)

All components can be ordered at http://reichelt.de

		Order Number	Description
D1	1	LED LL 5-8000RGBRGB-LED	5 mm, bedrahtet, 4-Pin, rt/gn/bl, 8000 mcd, 25°
SW1	1	TASTER 3301	Kurzhubtaster 6x6mm, Höhe: 4,3mm, 12V, vertikal
U1	2	MPE 087-1-008	Stiftleisten 2,54 mm, 1X08, gerade
R4	1	1/4W 10K	Widerstand, Kohleschicht, 10 kOhm, 0207, 250 mW, 5
R1-3	3	1/4W 300	Widerstand, Kohleschicht, 300 Ohm, 0207, 250 mW, 5%
	1	ESD BEUTEL-S 127	ESD PE-Abschirmbeutel, wiederverschließbar, 76 x 127 mm

Schematic

Assembly

1. Bend the wires of the resistors by 90°
   
2. Put the resistors through the holes (R4 is 10k while R1-3 is 330 Ohms)
   

3. Bend the wires by 30° and cut the pins, so 1mm wire is still visible from the bottom

4. Put the tactile button through the holes as described in following picture

5. Put the LEDs through the holes with the flatted side pointed in direction of the push button
6. Bend the outer wires by 30° and cut all wires, so 1mm wire is still visible from the bottom

7. Solder all components

8. Solder the pin headers (male) downwards

9. Solder the pin headers (female) on top of D1 mini, so the shield can be put on top.

Flashing firmware

See the latest release at the release page of Github!

Install Firmware:

```
pip3 install esptool
python3 -m esptool --port /dev/tty.usbserial-1440 --baud 115200
erase_flash
python3 -m esptool --port /dev/tty.SLAB_USBtoUART --baud 115200
write_flash 0x0 firmware-v1.0-esp8266_learning-iot.bin
```

WiFi-AP:
SSID: Accso Learning IoT
Key: Accso Learning IoT

URL
http://192.168.4.1
http://accso-learning-iot.local

Config-Area:
User: admin
Password: admin

SW API

Initialize Board

To initialize the Learning IoT Board requires as parameter the pointer to the web server handling REST-API calls.

```
LIoTBoard.begin(&webServer);
```

Setting LEDs

```
void LIoTBoard.setLed(String color);

color: Color as string. Can be:
 - "black"
 - "red"
 - "green"
 - "yellow"
 - "blue"
 - "purple"
 - "cyan"
 - "white"
```

```
void LIoTBoard.setLed(bool bRed, bool bGreen, bool bBlue);

bRed: LED red true = on, false = off

bGreen: LED green true = on, false = off

bBlue: LED blue true = on, false = off
```

Read button

```
bool LIoTBoard.getButtonPressed(void);

returns: true if button pressed and false if button is released
```

Getting started

Driving LED without board support

basic.ino

```
#define LED_RED    16 // D0
#define LED_GREEN   2 // D4
#define LED_BLUE   15 // D8
#define BUTTON      0 // D3

void setup(void)
{
    pinMode(LED_RED,OUTPUT);
    pinMode(LED_GREEN,OUTPUT);
    pinMode(LED_BLUE,OUTPUT);
```

```
    pinMode(BUTTON,INPUT);
}

void loop(void)
{
    // blink red LED
    digitalWrite(LED_RED,HIGH);
    delay(500);
    digitalWrite(LED_RED,LOW);
    delay(500);

    // blink green LED
    digitalWrite(LED_GREEN,HIGH);
    delay(500);
    digitalWrite(LED_GREEN,LOW);
    delay(500);

    // blink blue LED
    digitalWrite(LED_BLUE,HIGH);
    delay(500);
    digitalWrite(LED_BLUE,LOW);
    delay(500);

    // wait for button pressed (blocking)
    while(digitalRead(BUTTON))
    {
        delay(1);
    }
}
```

Driving LED with board support

board-support.ino

```
# requires to add liotboard.h and liotboard.cpp to the project
#include "liotboard.h"

void setup(void)
{
    LIoTBoard.begin();

    LIoTBoard.setLed("red");
}

void loop(void)
{
    LIoTBoard.update();
    if (LIoTBoard.getButtonPressed())
    {
        LIoTBoard.setLed("green");
    } else
```

```
    {
        LIoTBoard.setLed("red");
    }
}
```

REST API Example

POST data with JSON content /api/status:

- "red": LED Red, allowed values: "true" or "false"
- "green": LED Green, allowed values: "true" or "false"
- "blue": LED Blue, allowed values: "true" or "false"
- "button": Button change overwrite, allowed values: "true" or "false"

GET data as JSON format /api/status:

- "red": LED Red, allowed values: "true" or "false"
- "green": LED Green, allowed values: "true" or "false"
- "blue": LED Blue, allowed values: "true" or "false"
- "button": Button status, allowed values: "true" or "false"

restapi.ino

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

# requires to add liotboard.h and liotboard.cpp to the project
#include "liotboard.h"

#ifndef STASSID
#define STASSID "your-ssid"
#define STAPSK "your-password"
#endif

const char* ssid = STASSID;
const char* password = STAPSK;

ESP8266WebServer server(80);


void handleRoot() {
  server.send(200, "text/plain", "hello from esp8266!\r\n");
}

void setup(void) {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");
```

```
  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  if (MDNS.begin("esp8266")) { Serial.println("MDNS responder started"); }

  LIoTBoard.begin(&server);
  server.on("/", handleRoot);

  server.begin();
  Serial.println("HTTP server started");
}

void loop(void) {
  server.handleClient();
  MDNS.update();
}
```