ESP8266

Martin Heckel

2021–05–18

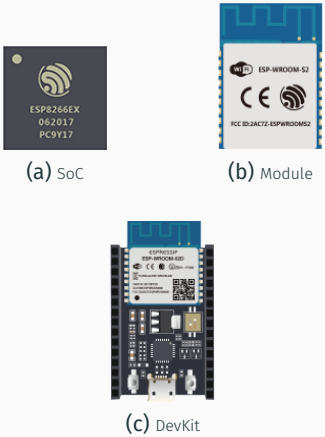
Hof University, University of Applied Sciences

Outline

Introduction

ESP8266?

- WiFi-capable MicroController Unit (MCU)
- Produced by Espressif Systems
- Can be programmed in different languages:
  - Arduino (similar to C++)
  - Python
  - Lua
  - C (for this presentation: RTOS-SDK [4])
  - ...
- Different form factors (see Figure 1)



(a) SoC (b) Module

(c) DevKit

Figure 1: ESP8266 in different configurations

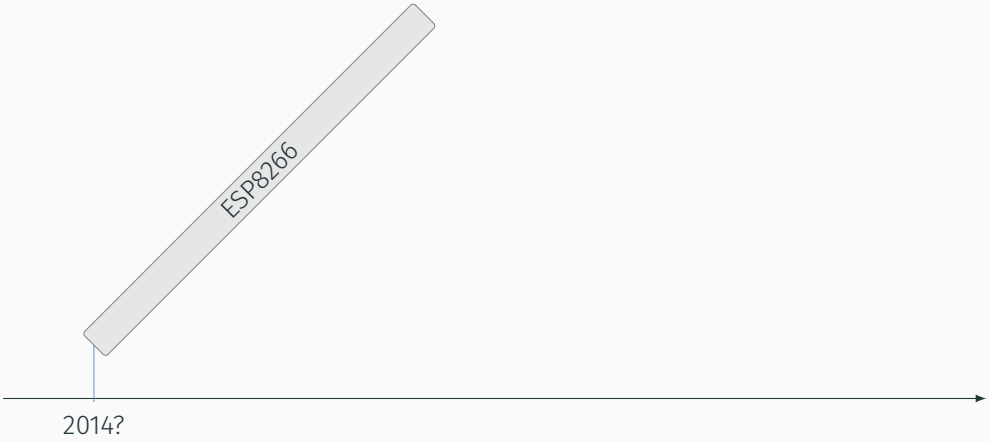
Images from Espressif SoCs and Espressif Modules

ESP32?

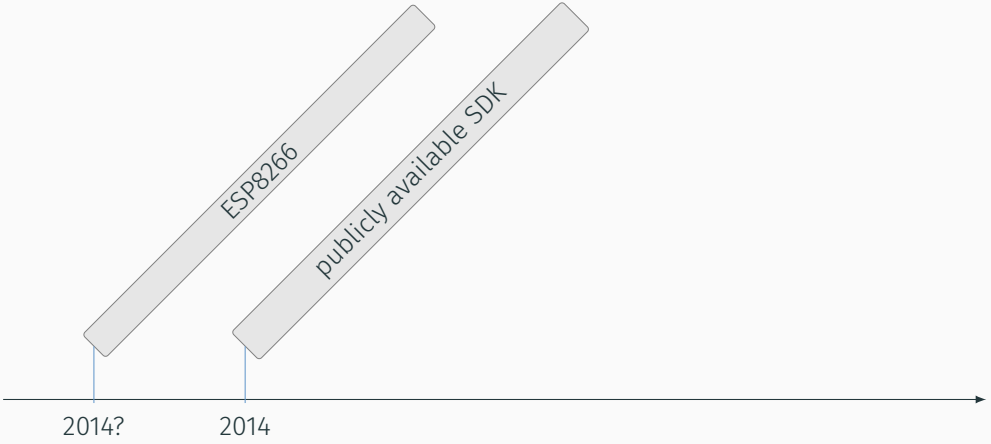
- Successor of the ESP8266
- Supports Bluetooth and Bluetooth Low Energy (BLE) additionally to WiFi

Timeline

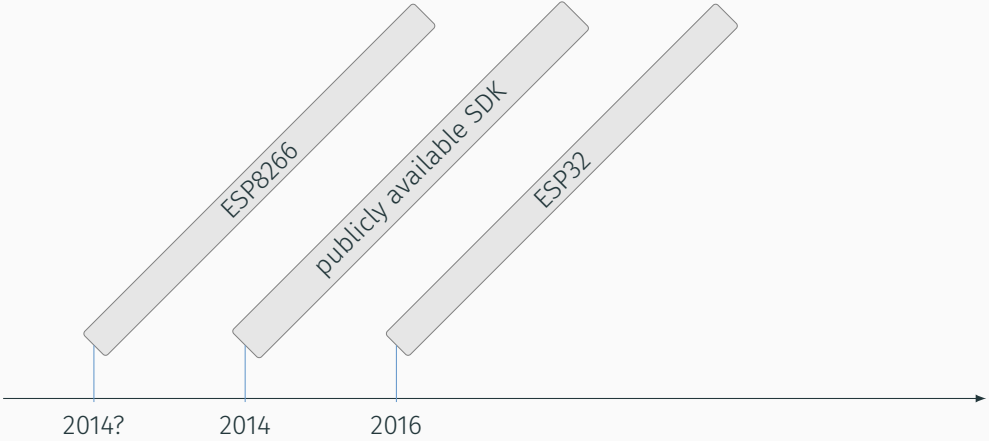
Timeline



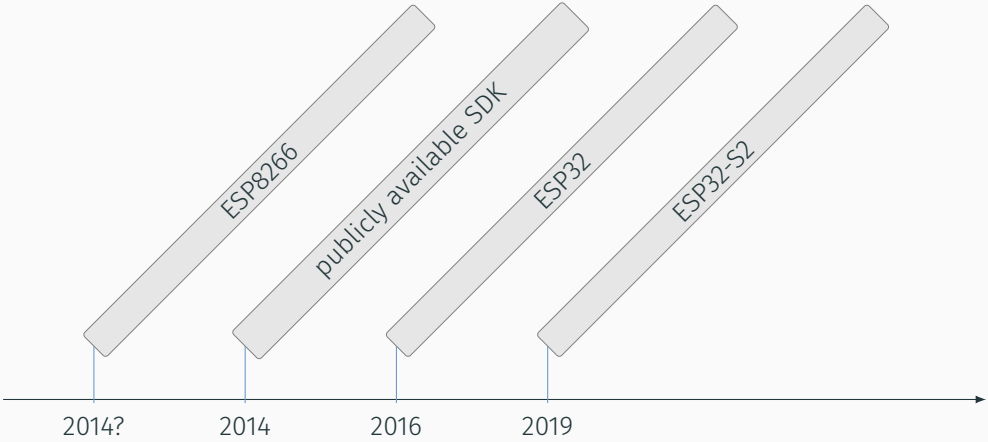
Timeline



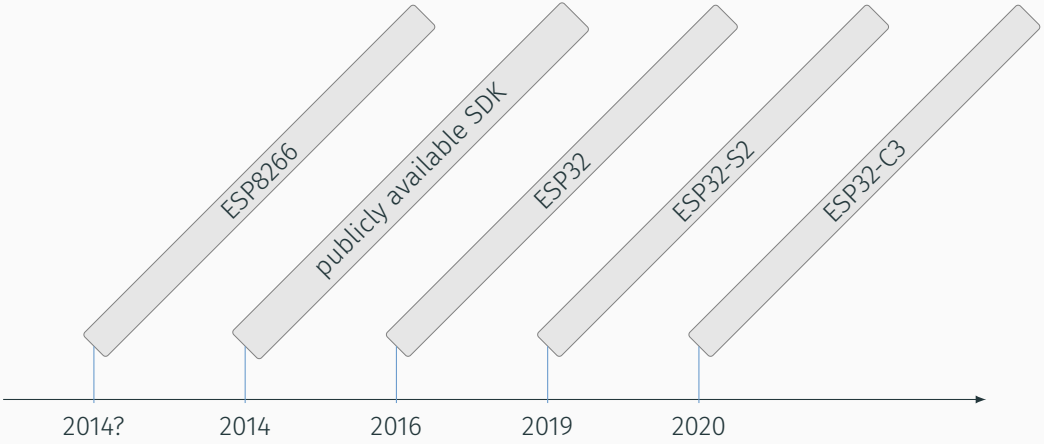
Timeline



Timeline


ESP8266
publicly available SDK
ESP32
ESP32-S2
2014?
2014
2016
2019

Timeline


ESP8266
publicly available SDK
ESP32
ESP32-S2
ESP32-C3
2014?
2014
2016
2019
2020

Comparison: Arduino vs ESP8266

Feature	Arduino Uno [1]	ESP8266 [3]	ESP32 [2]
WiFi	✗	802.11 b/g/n	802.11 b/g/n
Bluetooth	✗	✗	BT 4.2 & BLE
GPIO	✓	✓	✓
CPU	8-bit, 16 MIPS with 16 MHz	32-bit, 160 MHz	32-bit, 200, 400, or 600 MIPS
Memory	2 KiB	160 KiB	520 KiB
Flash	32 KiB	up to 16 MiB external flash	up to 16 MiB external flash
Power consumption	4.5 mW at 4 MHz	264 mW average	500 mW average

Programming examples

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

```
1   void app_main(void) {
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

GPIOs and physical ports

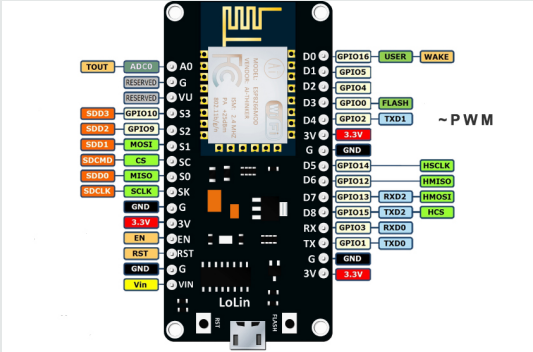


Image from teachmemicro.com

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

GPIO pins – General usage

```
1  void app_main(void) {
2    setupGPIO();
3
4    ESP_LOGI("main", "Started.");
5    int level;
6
7    while(1) {
8      level = gpio_get_level(4);
9      if(level == 0) {
10       ESP_LOGI("main", "Blinking.");
11       gpio_set_level(5, 1);
12       sys_delay_ms(500);
13       gpio_set_level(5, 0);
14       sys_delay_ms(500);
15     } else {
16       sys_delay_ms(10);
17     }
18   }
19 }
```

GPIO pins – Setup

```
1  void setupGPIO(void) {
2    gpio_config_t config;
3
4    config.pin_bit_mask = (1<<5);
5    config.mode = GPIO_MODE_OUTPUT;
6    config.pull_up_en = GPIO_PULLUP_DISABLE;
7
8    gpio_config(&config);
9
10   config.pin_bit_mask = (1<<4);
11   config.mode = GPIO_MODE_INPUT;
12   config.pull_up_en = GPIO_PULLUP_ENABLE;
13
14   gpio_config(&config);
15 }
```

GPIO pins – Setup

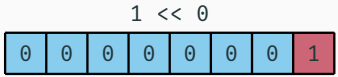
```
1  void setupGPIO(void) {
2    gpio_config_t config;
3
4    config.pin_bit_mask = (1<<5);
5    config.mode = GPIO_MODE_OUTPUT;
6    config.pull_up_en = GPIO_PULLUP_DISABLE;
7
8    gpio_config(&config);
9
10   config.pin_bit_mask = (1<<4);
11   config.mode = GPIO_MODE_INPUT;
12   config.pull_up_en = GPIO_PULLUP_ENABLE;
13
14   gpio_config(&config);
15 }
```

GPIO pins – Setup

```
1  void setupGPIO(void) {
2    gpio_config_t config;
3
4    config.pin_bit_mask = (1<<5);
5    config.mode = GPIO_MODE_OUTPUT;
6    config.pull_up_en = GPIO_PULLUP_DISABLE;
7
8    gpio_config(&config);
9
10   config.pin_bit_mask = (1<<4);
11   config.mode = GPIO_MODE_INPUT;
12   config.pull_up_en = GPIO_PULLUP_ENABLE;
13
14   gpio_config(&config);
15 }
```

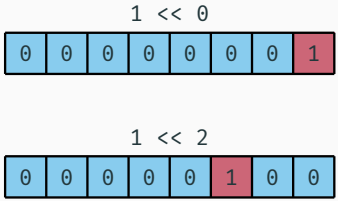
GPIO pins – Setup

```
1  void setupGPIO(void) {
2    gpio_config_t config;
3
4    config.pin_bit_mask = (1<<5);
5    config.mode = GPIO_MODE_OUTPUT;
6    config.pull_up_en = GPIO_PULLUP_DISABLE;
7
8    gpio_config(&config);
9
10   config.pin_bit_mask = (1<<4);
11   config.mode = GPIO_MODE_INPUT;
12   config.pull_up_en = GPIO_PULLUP_ENABLE;
13
14   gpio_config(&config);
15 }
```



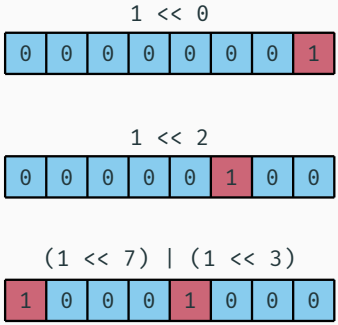
GPIO pins – Setup

```
1  void setupGPIO(void) {
2    gpio_config_t config;
3
4    config.pin_bit_mask = (1<<5);
5    config.mode = GPIO_MODE_OUTPUT;
6    config.pull_up_en = GPIO_PULLUP_DISABLE;
7
8    gpio_config(&config);
9
10   config.pin_bit_mask = (1<<4);
11   config.mode = GPIO_MODE_INPUT;
12   config.pull_up_en = GPIO_PULLUP_ENABLE;
13
14   gpio_config(&config);
15 }
```



GPIO pins – Setup

```
1  void setupGPIO(void) {
2    gpio_config_t config;
3
4    config.pin_bit_mask = (1<<5);
5    config.mode = GPIO_MODE_OUTPUT;
6    config.pull_up_en = GPIO_PULLUP_DISABLE;
7
8    gpio_config(&config);
9
10   config.pin_bit_mask = (1<<4);
11   config.mode = GPIO_MODE_INPUT;
12   config.pull_up_en = GPIO_PULLUP_ENABLE;
13
14   gpio_config(&config);
15 }
```



GPIO pins – Setup

```
1  void setupGPIO(void) {
2    gpio_config_t config;
3
4    config.pin_bit_mask = (1<<5);
5    config.mode = GPIO_MODE_OUTPUT;
6    config.pull_up_en = GPIO_PULLUP_DISABLE;
7
8    gpio_config(&config);
9
10   config.pin_bit_mask = (1<<4);
11   config.mode = GPIO_MODE_INPUT;
12   config.pull_up_en = GPIO_PULLUP_ENABLE;
13
14   gpio_config(&config);
15 }
```

GPIO pins – Setup

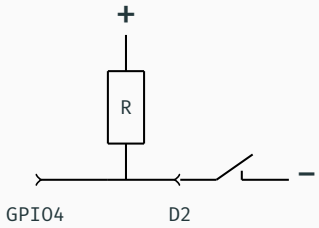
```
1  void setupGPIO(void) {
2    gpio_config_t config;
3
4    config.pin_bit_mask = (1<<5);
5    config.mode = GPIO_MODE_OUTPUT;
6    config.pull_up_en = GPIO_PULLUP_DISABLE;
7
8    gpio_config(&config);
9
10   config.pin_bit_mask = (1<<4);
11   config.mode = GPIO_MODE_INPUT;
12   config.pull_up_en = GPIO_PULLUP_ENABLE;
13
14   gpio_config(&config);
15 }
```

GPIO pins – Setup

```
1  void setupGPIO(void) {
2    gpio_config_t config;
3
4    config.pin_bit_mask = (1<<5);
5    config.mode = GPIO_MODE_OUTPUT;
6    config.pull_up_en = GPIO_PULLUP_DISABLE;
7
8    gpio_config(&config);
9
10   config.pin_bit_mask = (1<<4);
11   config.mode = GPIO_MODE_INPUT;
12   config.pull_up_en = GPIO_PULLUP_ENABLE;
13
14   gpio_config(&config);
15 }
```

GPIO pins – Setup

```
1  void setupGPIO(void) {
2    gpio_config_t config;
3
4    config.pin_bit_mask = (1<<5);
5    config.mode = GPIO_MODE_OUTPUT;
6    config.pull_up_en = GPIO_PULLUP_DISABLE;
7
8    gpio_config(&config);
9
10   config.pin_bit_mask = (1<<4);
11   config.mode = GPIO_MODE_INPUT;
12   config.pull_up_en = GPIO_PULLUP_ENABLE;
13
14   gpio_config(&config);
15 }
```

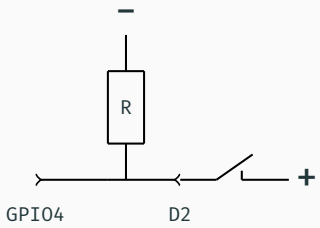


GPIO pins – Setup

```
1   void setupGPIO(void) {
2     gpio_config_t config;
3
4     config.pin_bit_mask = (1<<5);
5     config.mode = GPIO_MODE_OUTPUT;
6     config.pull_up_en = GPIO_PULLUP_DISABLE;
7
8     gpio_config(&config);
9
10    config.pin_bit_mask = (1<<4);
11    config.mode = GPIO_MODE_INPUT;
12    config.pull_up_en = GPIO_PULLUP_ENABLE;
13
14    gpio_config(&config);
15  }
```

GPIO pins – Setup

```
1   void setupGPIO(void) {
2     gpio_config_t config;
3
4     config.pin_bit_mask = (1<<5);
5     config.mode = GPIO_MODE_OUTPUT;
6     config.pull_up_en = GPIO_PULLUP_DISABLE;
7
8     gpio_config(&config);
9
10    config.pin_bit_mask = (1<<4);
11    config.mode = GPIO_MODE_INPUT;
12    config.pull_up_en = GPIO_PULLUP_ENABLE;
13
14    gpio_config(&config);
15  }
```



GPIO pins – Setup

```
1  void setupGPIO(void) {
2    gpio_config_t config;
3
4    config.pin_bit_mask = (1<<5);
5    config.mode = GPIO_MODE_OUTPUT;
6    config.pull_up_en = GPIO_PULLUP_DISABLE;
7
8    gpio_config(&config);
9
10   config.pin_bit_mask = (1<<4);
11   config.mode = GPIO_MODE_INPUT;
12   config.pull_up_en = GPIO_PULLUP_ENABLE;
13
14   gpio_config(&config);
15 }
```

Demo 1

GPIO example

WiFi Station – Setup

```
void initWifi() {
  tcpip_adapter_init();
  esp_event_loop_create_default();

  wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
  esp_wifi_init((const wifi_init_config_t *)(&cnf));

  esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
  esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);

  wifi_config_t wcnf;
  setIntArr(wcnf.sta.ssid, "myssid");
  setIntArr(wcnf.sta.password, "mypassword");
  wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;

  esp_wifi_set_mode(WIFI_MODE_STA);
  esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
  esp_wifi_start();
}
```

WiFi Station – Setup

```
1  void initWifi() {
2    tcpip_adapter_init();
3    esp_event_loop_create_default();
4
5    wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
6    esp_wifi_init((const wifi_init_config_t *)(&cnf));
7
8    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
9    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);
10
11   wifi_config_t wcnf;
12   setIntArr(wcnf.sta.ssid, "myssid");
13   setIntArr(wcnf.sta.password, "mypassword");
14   wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;
15
16   esp_wifi_set_mode(WIFI_MODE_STA);
17   esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
18   esp_wifi_start();
19 }
```

WiFi Station – Setup

```
1  void initWifi() {
2    tcpip_adapter_init();
3    esp_event_loop_create_default();
4
5    wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
6    esp_wifi_init((const wifi_init_config_t *)(&cnf));
7
8    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
9    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);
10
11   wifi_config_t wcnf;
12   setIntArr(wcnf.sta.ssid, "myssid");
13   setIntArr(wcnf.sta.password, "mypassword");
14   wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;
15
16   esp_wifi_set_mode(WIFI_MODE_STA);
17   esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
18   esp_wifi_start();
19 }
```

WiFi Station – Setup

```
1  void initWifi() {
2    tcpip_adapter_init();
3    esp_event_loop_create_default();
4
5    wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
6    esp_wifi_init((const wifi_init_config_t *)(&cnf));
7
8    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
9    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);
10
11   wifi_config_t wcnf;
12   setIntArr(wcnf.sta.ssid, "myssid");
13   setIntArr(wcnf.sta.password, "mypassword");
14   wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;
15
16   esp_wifi_set_mode(WIFI_MODE_STA);
17   esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
18   esp_wifi_start();
19 }
```

WiFi Station – Setup

```
1  void initWifi() {
2    tcpip_adapter_init();
3    esp_event_loop_create_default();
4
5    wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
6    esp_wifi_init((const wifi_init_config_t *)(&cnf));
7
8    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
9    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);
10
11   wifi_config_t wcnf;
12   setIntArr(wcnf.sta.ssid, "myssid");
13   setIntArr(wcnf.sta.password, "mypassword");
14   wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;
15
16   esp_wifi_set_mode(WIFI_MODE_STA);
17   esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
18   esp_wifi_start();
19 }
```

WiFi Station – Setup

```
1  void initWifi() {
2    tcpip_adapter_init();
3    esp_event_loop_create_default();
4
5    wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
6    esp_wifi_init((const wifi_init_config_t *)(&cnf));
7
8    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
9    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);
10
11   wifi_config_t wcnf;
12   setIntArr(wcnf.sta.ssid, "myssid");
13   setIntArr(wcnf.sta.password, "mypassword");
14   wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;
15
16   esp_wifi_set_mode(WIFI_MODE_STA);
17   esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
18   esp_wifi_start();
19 }
```

WiFi Station – Setup

```
1  void initWifi() {
2    tcpip_adapter_init();
3    esp_event_loop_create_default();
4
5    wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
6    esp_wifi_init((const wifi_init_config_t *)(&cnf));
7
8    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
9    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);
10
11   wifi_config_t wcnf;
12   setIntArr(wcnf.sta.ssid, "myssid");
13   setIntArr(wcnf.sta.password, "mypassword");
14   wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;
15
16   esp_wifi_set_mode(WIFI_MODE_STA);
17   esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
18   esp_wifi_start();
19 }
```

WiFi Station – Setup

```
1  void initWifi() {
2    tcpip_adapter_init();
3    esp_event_loop_create_default();
4
5    wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
6    esp_wifi_init((const wifi_init_config_t *)(&cnf));
7
8    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
9    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);
10
11   wifi_config_t wcnf;
12   setIntArr(wcnf.sta.ssid, "myssid");
13   setIntArr(wcnf.sta.password, "mypassword");
14   wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;
15
16   esp_wifi_set_mode(WIFI_MODE_STA);
17   esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
18   esp_wifi_start();
19 }
```

WiFi Station – Setup

```
1  void initWifi() {
2    tcpip_adapter_init();
3    esp_event_loop_create_default();
4
5    wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
6    esp_wifi_init((const wifi_init_config_t *)(&cnf));
7
8    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
9    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);
10
11   wifi_config_t wcnf;
12   setIntArr(wcnf.sta.ssid, "myssid");
13   setIntArr(wcnf.sta.password, "mypassword");
14   wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;
15
16   esp_wifi_set_mode(WIFI_MODE_STA);
17   esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
18   esp_wifi_start();
19 }
```

WiFi Station – Setup

```
1  void initWifi() {
2    tcpip_adapter_init();
3    esp_event_loop_create_default();
4
5    wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
6    esp_wifi_init((const wifi_init_config_t *)(&cnf));
7
8    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
9    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);
10
11   wifi_config_t wcnf;
12   setIntArr(wcnf.sta.ssid, "myssid");
13   setIntArr(wcnf.sta.password, "mypassword");
14   wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;
15
16   esp_wifi_set_mode(WIFI_MODE_STA);
17   esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
18   esp_wifi_start();
19 }
```

WiFi Station – Setup

```
1  void initWifi() {
2    tcpip_adapter_init();
3    esp_event_loop_create_default();
4
5    wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
6    esp_wifi_init((const wifi_init_config_t *)(&cnf));
7
8    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
9    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);
10
11   wifi_config_t wcnf;
12   setIntArr(wcnf.sta.ssid, "myssid");
13   setIntArr(wcnf.sta.password, "mypassword");
14   wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;
15
16   esp_wifi_set_mode(WIFI_MODE_STA);
17   esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
18   esp_wifi_start();
19 }
```

WiFi Station – Setup

```
1  void initWifi() {
2    tcpip_adapter_init();
3    esp_event_loop_create_default();
4
5    wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
6    esp_wifi_init((const wifi_init_config_t *)(&cnf));
7
8    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
9    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);
10
11   wifi_config_t wcnf;
12   setIntArr(wcnf.sta.ssid, "myssid");
13   setIntArr(wcnf.sta.password, "mypassword");
14   wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;
15
16   esp_wifi_set_mode(WIFI_MODE_STA);
17   esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
18   esp_wifi_start();
19 }
```

WiFi Station – Setup

```
1  void initWifi() {
2    tcpip_adapter_init();
3    esp_event_loop_create_default();
4
5    wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
6    esp_wifi_init((const wifi_init_config_t *)(&cnf));
7
8    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
9    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);
10
11   wifi_config_t wcnf;
12   setIntArr(wcnf.sta.ssid, "myssid");
13   setIntArr(wcnf.sta.password, "mypassword");
14   wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;
15
16   esp_wifi_set_mode(WIFI_MODE_STA);
17   esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
18   esp_wifi_start();
19 }
```

WiFi Station – Setup

```
1  void initWifi() {
2    tcpip_adapter_init();
3    esp_event_loop_create_default();
4
5    wifi_init_config_t cnf = WIFI_INIT_CONFIG_DEFAULT();
6    esp_wifi_init((const wifi_init_config_t *)(&cnf));
7
8    esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &handler, NULL);
9    esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &handler, NULL);
10
11   wifi_config_t wcnf;
12   setIntArr(wcnf.sta.ssid, "myssid");
13   setIntArr(wcnf.sta.password, "mypassword");
14   wcnf.sta.threshold.authmode = WIFI_AUTH_WPA2_PSK;
15
16   esp_wifi_set_mode(WIFI_MODE_STA);
17   esp_wifi_set_config(ESP_IF_WIFI_STA, &wcnf);
18   esp_wifi_start();
19 }
```

WiFi Station – Event handling

```
void handler(void *arg, esp_event_base_t base, int32_t eventId, void *data) {
  if(base == WIFI_EVENT && eventId == WIFI_EVENT_STA_START) {
    ESP_LOGI("WiFi", "Connecting to AP...");
    esp_wifi_connect();
  }
  else if (base == WIFI_EVENT && eventId == WIFI_EVENT_STA_DISCONNECTED) {
    wifi_event_sta_disconnected_t *event = (wifi_event_sta_disconnected_t*)data;
    ESP_LOGI("WiFi", "[%d] Disconnected. Reconnecting...", event->reason);
    esp_wifi_connect();
  }
  else if(base == IP_EVENT && eventId == IP_EVENT_STA_GOT_IP) {
    ip_event_got_ip_t *event = (ip_event_got_ip_t *)data;
    ESP_LOGI("WiFi", "Connected to AP. IP=%s", ip4addr_ntoa(&event->ip_info.ip));
  }
}
```

WiFi Station – Event handling

```
1  void handler(void *arg, esp_event_base_t base, int32_t eventId, void *data) {
2    if(base == WIFI_EVENT && eventId == WIFI_EVENT_STA_START) {
3      ESP_LOGI("WiFi", "Connecting to AP...");
4      esp_wifi_connect();
5    }
6    else if (base == WIFI_EVENT && eventId == WIFI_EVENT_STA_DISCONNECTED) {
7      wifi_event_sta_disconnected_t *event = (wifi_event_sta_disconnected_t*)data;
8      ESP_LOGI("WiFi", "[%d] Disconnected. Reconnecting...", event->reason);
9      esp_wifi_connect();
10   }
11   else if(base == IP_EVENT && eventId == IP_EVENT_STA_GOT_IP) {
12     ip_event_got_ip_t *event = (ip_event_got_ip_t *)data;
13     ESP_LOGI("WiFi", "Connected to AP. IP=%s", ip4addr_ntoa(&event->ip_info.ip));
14   }
15 }
```

WiFi Station – Event handling

```
1  void handler(void *arg, esp_event_base_t base, int32_t eventId, void *data) {
2    if(base == WIFI_EVENT && eventId == WIFI_EVENT_STA_START) {
3      ESP_LOGI("WiFi", "Connecting to AP...");
4      esp_wifi_connect();
5    }
6    else if (base == WIFI_EVENT && eventId == WIFI_EVENT_STA_DISCONNECTED) {
7      wifi_event_sta_disconnected_t *event = (wifi_event_sta_disconnected_t*)data;
8      ESP_LOGI("WiFi", "[%d] Disconnected. Reconnecting...", event->reason);
9      esp_wifi_connect();
10   }
11   else if(base == IP_EVENT && eventId == IP_EVENT_STA_GOT_IP) {
12     ip_event_got_ip_t *event = (ip_event_got_ip_t *)data;
13     ESP_LOGI("WiFi", "Connected to AP. IP=%s", ip4addr_ntoa(&event->ip_info.ip));
14   }
15 }
```

WiFi Station – Event handling

```
1  void handler(void *arg, esp_event_base_t base, int32_t eventId, void *data) {
2    if(base == WIFI_EVENT && eventId == WIFI_EVENT_STA_START) {
3      ESP_LOGI("WiFi", "Connecting to AP...");
4      esp_wifi_connect();
5    }
6    else if (base == WIFI_EVENT && eventId == WIFI_EVENT_STA_DISCONNECTED) {
7      wifi_event_sta_disconnected_t *event = (wifi_event_sta_disconnected_t*)data;
8      ESP_LOGI("WiFi", "[%d] Disconnected. Reconnecting...", event->reason);
9      esp_wifi_connect();
10   }
11   else if(base == IP_EVENT && eventId == IP_EVENT_STA_GOT_IP) {
12     ip_event_got_ip_t *event = (ip_event_got_ip_t *)data;
13     ESP_LOGI("WiFi", "Connected to AP. IP=%s", ip4addr_ntoa(&event->ip_info.ip));
14   }
15 }
```

WiFi Station – Event handling

```
1  void handler(void *arg, esp_event_base_t base, int32_t eventId, void *data) {
2    if(base == WIFI_EVENT && eventId == WIFI_EVENT_STA_START) {
3      ESP_LOGI("WiFi", "Connecting to AP...");
4      esp_wifi_connect();
5    }
6    else if (base == WIFI_EVENT && eventId == WIFI_EVENT_STA_DISCONNECTED) {
7      wifi_event_sta_disconnected_t *event = (wifi_event_sta_disconnected_t*)data;
8      ESP_LOGI("WiFi", "[%d] Disconnected. Reconnecting...", event->reason);
9      esp_wifi_connect();
10   }
11   else if(base == IP_EVENT && eventId == IP_EVENT_STA_GOT_IP) {
12     ip_event_got_ip_t *event = (ip_event_got_ip_t *)data;
13     ESP_LOGI("WiFi", "Connected to AP. IP=%s", ip4addr_ntoa(&event->ip_info.ip));
14   }
15 }
```

WiFi Station – Event handling

```
1  void handler(void *arg, esp_event_base_t base, int32_t eventId, void *data) {
2    if(base == WIFI_EVENT && eventId == WIFI_EVENT_STA_START) {
3      ESP_LOGI("WiFi", "Connecting to AP...");
4      esp_wifi_connect();
5    }
6    else if (base == WIFI_EVENT && eventId == WIFI_EVENT_STA_DISCONNECTED) {
7      wifi_event_sta_disconnected_t *event = (wifi_event_sta_disconnected_t*)data;
8      ESP_LOGI("WiFi", "[%d] Disconnected. Reconnecting...", event->reason);
9      esp_wifi_connect();
10   }
11   else if(base == IP_EVENT && eventId == IP_EVENT_STA_GOT_IP) {
12     ip_event_got_ip_t *event = (ip_event_got_ip_t *)data;
13     ESP_LOGI("WiFi", "Connected to AP. IP=%s", ip4addr_ntoa(&event->ip_info.ip));
14   }
15 }
```

WiFi Station – Event handling

```
1  void handler(void *arg, esp_event_base_t base, int32_t eventId, void *data) {
2    if(base == WIFI_EVENT && eventId == WIFI_EVENT_STA_START) {
3      ESP_LOGI("WiFi", "Connecting to AP...");
4      esp_wifi_connect();
5    }
6    else if (base == WIFI_EVENT && eventId == WIFI_EVENT_STA_DISCONNECTED) {
7      wifi_event_sta_disconnected_t *event = (wifi_event_sta_disconnected_t*)data;
8      ESP_LOGI("WiFi", "[%d] Disconnected. Reconnecting...", event->reason);
9      esp_wifi_connect();
10   }
11   else if(base == IP_EVENT && eventId == IP_EVENT_STA_GOT_IP) {
12     ip_event_got_ip_t *event = (ip_event_got_ip_t *)data;
13     ESP_LOGI("WiFi", "Connected to AP. IP=%s", ip4addr_ntoa(&event->ip_info.ip));
14   }
15 }
```

WiFi Station – Event handling

```
1  void handler(void *arg, esp_event_base_t base, int32_t eventId, void *data) {
2    if(base == WIFI_EVENT && eventId == WIFI_EVENT_STA_START) {
3      ESP_LOGI("WiFi", "Connecting to AP...");
4      esp_wifi_connect();
5    }
6    else if (base == WIFI_EVENT && eventId == WIFI_EVENT_STA_DISCONNECTED) {
7      wifi_event_sta_disconnected_t *event = (wifi_event_sta_disconnected_t*)data;
8      ESP_LOGI("WiFi", "[%d] Disconnected. Reconnecting...", event->reason);
9      esp_wifi_connect();
10   }
11   else if(base == IP_EVENT && eventId == IP_EVENT_STA_GOT_IP) {
12     ip_event_got_ip_t *event = (ip_event_got_ip_t *)data;
13     ESP_LOGI("WiFi", "Connected to AP. IP=%s", ip4addr_ntoa(&event->ip_info.ip));
14   }
15 }
```

WiFi Station – Event handling

```
1  void handler(void *arg, esp_event_base_t base, int32_t eventId, void *data) {
2    if(base == WIFI_EVENT && eventId == WIFI_EVENT_STA_START) {
3      ESP_LOGI("WiFi", "Connecting to AP...");
4      esp_wifi_connect();
5    }
6    else if (base == WIFI_EVENT && eventId == WIFI_EVENT_STA_DISCONNECTED) {
7      wifi_event_sta_disconnected_t *event = (wifi_event_sta_disconnected_t*)data;
8      ESP_LOGI("WiFi", "[%d] Disconnected. Reconnecting...", event->reason);
9      esp_wifi_connect();
10   }
11   else if(base == IP_EVENT && eventId == IP_EVENT_STA_GOT_IP) {
12     ip_event_got_ip_t *event = (ip_event_got_ip_t *)data;
13     ESP_LOGI("WiFi", "Connected to AP. IP=%s", ip4addr_ntoa(&event->ip_info.ip));
14   }
15 }
```

WiFi Station – Event handling

```
1  void handler(void *arg, esp_event_base_t base, int32_t eventId, void *data) {
2    if(base == WIFI_EVENT && eventId == WIFI_EVENT_STA_START) {
3      ESP_LOGI("WiFi", "Connecting to AP...");
4      esp_wifi_connect();
5    }
6    else if (base == WIFI_EVENT && eventId == WIFI_EVENT_STA_DISCONNECTED) {
7      wifi_event_sta_disconnected_t *event = (wifi_event_sta_disconnected_t*)data;
8      ESP_LOGI("WiFi", "[%d] Disconnected. Reconnecting...", event->reason);
9      esp_wifi_connect();
10   }
11   else if(base == IP_EVENT && eventId == IP_EVENT_STA_GOT_IP) {
12     ip_event_got_ip_t *event = (ip_event_got_ip_t *)data;
13     ESP_LOGI("WiFi", "Connected to AP. IP=%s", ip4addr_ntoa(&event->ip_info.ip));
14   }
15 }
```

WiFi Station – Event handling

```
1  void handler(void *arg, esp_event_base_t base, int32_t eventId, void *data) {
2    if(base == WIFI_EVENT && eventId == WIFI_EVENT_STA_START) {
3      ESP_LOGI("WiFi", "Connecting to AP...");
4      esp_wifi_connect();
5    }
6    else if (base == WIFI_EVENT && eventId == WIFI_EVENT_STA_DISCONNECTED) {
7      wifi_event_sta_disconnected_t *event = (wifi_event_sta_disconnected_t*)data;
8      ESP_LOGI("WiFi", "[%d] Disconnected. Reconnecting...", event->reason);
9      esp_wifi_connect();
10   }
11   else if(base == IP_EVENT && eventId == IP_EVENT_STA_GOT_IP) {
12     ip_event_got_ip_t *event = (ip_event_got_ip_t *)data;
13     ESP_LOGI("WiFi", "Connected to AP. IP=%s", ip4addr_ntoa(&event->ip_info.ip));
14   }
15 }
```

WiFi Station – Event handling

```
1  void handler(void *arg, esp_event_base_t base, int32_t eventId, void *data) {
2    if(base == WIFI_EVENT && eventId == WIFI_EVENT_STA_START) {
3      ESP_LOGI("WiFi", "Connecting to AP...");
4      esp_wifi_connect();
5    }
6    else if (base == WIFI_EVENT && eventId == WIFI_EVENT_STA_DISCONNECTED) {
7      wifi_event_sta_disconnected_t *event = (wifi_event_sta_disconnected_t*)data;
8      ESP_LOGI("WiFi", "[%d] Disconnected. Reconnecting...", event->reason);
9      esp_wifi_connect();
10   }
11   else if(base == IP_EVENT && eventId == IP_EVENT_STA_GOT_IP) {
12     ip_event_got_ip_t *event = (ip_event_got_ip_t *)data;
13     ESP_LOGI("WiFi", "Connected to AP. IP=%s", ip4addr_ntoa(&event->ip_info.ip));
14   }
15 }
```

Demo 2

Connecting to a WiFi AP as Station

WiFi AP

- Basically the same as WiFi station
  - Initialize tcpip adapter
  - Implement event handlers and register them
  - Create and initialize configuration data structure
  - Set mode to Access point
  - Start WiFi

Webserver

- Initialize webserver
- Add handlers for different methods and paths
  - Load the query string to a buffer
  - Do some application logic (e.g. checking if a parameter has a specified value, blink a LED, etc.)
  - Send response
- Start the server

Demo 3

Controlling LED over WiFi

Aspects of information security

How trustworthy is the ESP?

- Trust in hardware that might not be trustworthy
- Bugs in SDKs, libs, etc. -> update frequency?
- Bugs in own implementation -> update frequency?

Idea: How to improve overall security

- **Basic approach**: Reduce attack surface
  - Put them in an additional network
  - Access from the outside only with a proxy
- **Awareness**: Be aware of the risks
  - Confidentiallity
  - Integrity
  - Availability

Project ideas related to IoT

Web-controllable LED-Strip

- Setup a webserver that processes requests
- Control a WS2812-based LED strip based on the submitted requests
- Works with matrices, cubes, etc. as well

Controller for 3D-Snake

- Part of the project “3D-Snake” created in the scope of the bachelor program
- Controllers have photoelectric barrier and detect motion based on them
- Movements are interpreted as gestures and submitted to the main component (Raspberry Pi) via Bluetooth
- Idea: Add other controller types (e.g. accelerometer, etc.)

Project ideas not related to IoT

WiFi Deauthentication

- Security of WiFi ist not as good as it should be
- There is the possibility to deauthenticate a client by spoofing its MAC address

WiFi Deauthentication

- **Idea**: Deauthenticate all clients from a given SSID
  - If a client connects, its connection is deauthenticated again
  - Impact on Availability

WiFi Deauthentication

- **Idea**: Deauthenticate all clients from a given SSID and provide a network with the same SSID
  - Clients connect to the attackers network
  - Impact on Integrity/Confidentiallity

WiFi Deauthentication

- **Idea**: Deauthenticate clients and let them connect again
  - Sniff the WPA 4-Way handshake
  - Crack the WPA key after enough handshakes are sniffed

Conclusion

Conclusion

- Small programmable device with WiFi capability
- Programmable in different languages
- Documentation from vendor not satisfying, but many 3rd party resources
- Think about risks related to security/safety of your projects

Questions?

References i

[1] *ATmega328P Datasheet*. 2015. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (visited on 03/24/2021).

[2] *ESP32 Datasheet*. 2021. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (visited on 03/24/2021).

[3] *ESP8266 Datasheet*. 2020. URL: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf (visited on 03/24/2021).

[4] *ESP8266 RTOS Software Development Kit*. 2021. URL: https://github.com/espressif/ESP8266_RTOS_SDK (visited on 05/16/2021).