Project Integration
Research Applied Computer Science

Jochem Arends
495637
Group 1

Academic Year: 2023-2024

Contents

Abbreviations

ACS	Applied Computer Science.
AIoT	Artificial Intelligence of Things.
HTTP	Hypertext Transfer Protocol.
SDK	Software Development Kit.

Glossary

Bluetooth	A short-range wireless interconnection of mobile phones, computers, and other electronic devices.
WiFi	A wireless networking technology that uses radio waves to provide high-speed Internet access.

1 Introduction

As part of Project Integration, Applied Computer Science (ACS) students need to conduct research about the ESP32 microcontroller. Upon completion of this research, a set of assignments have to be completed. First, some introductory assignments in order to familiarize ourselves with concepts that are important for the project, followed by a final assignment that goes more in depth into specific topics that are of value for our final product.

2 The ESP32

The ESP32 is a series of microcontrollers developed by Espressif Systems that have integrated WiFi and Bluetooth modules (Espressif, 2024).

3 Espressif

Espressif Systems is a multinational semiconductor company that focusses among other things on developing wireless communication and Artificial Intelligence of Things (AIoT) (Espressif, n.d.). Popular products Espressif created, are the ESP32 series of chips, development boards, and modules (Espressif, n.d.). Espressif supports various open-source software projects such as Software Development Kits (SDKs), libraries, and tools. Espressif is a chinese company but has offices all around the world (Espressif, n.d.).

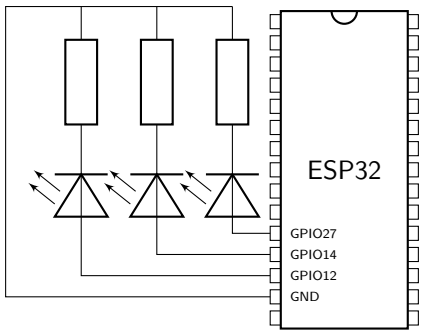
4 Lilygo

5 Assignment 1

Show that the ESP32 can turn three different LEDs on and off seperately using an internal loop with delays.

5.1 Circuit

For this assignment, the circuit used is quite straightforward. Connect three LEDs to the ESP32, put these LEDs in series with a resistor, and connect them to a common ground. For all subsequent assignments, it can be assumed that the same circuit is used unless explicitly stated otherwise.



5.2 Toolchain

For this assignment, I decided on using the ESP-IDF toolchain. More specifically, I used the `idf.py` command line tool in combination with a text editor. I never used the ESP-IDF toolchain before and was curious about what it would be like. One thing I really like about the ESP-IDF toolchain is that it provides a C++ compiler that supports both language and library features of more recent C++ standards, which is something the Arduino IDE lacks.

5.3 Software

```
#include <array>
#include <chrono>
#include <ranges>
#include <thread>

#include "driver/gpio.h"
#include "rom/gpio.h"

extern "C" void app_main() {
    const std::array<gpio_num_t, 3> led_pins{
        GPIO_NUM_12,
        GPIO_NUM_14,
        GPIO_NUM_27,
    };

    // configure the pins for output
    for (auto pin : led_pins) {
        gpio_pad_select_gpio(pin);
        gpio_reset_pin(pin);
        gpio_set_direction(pin, GPIO_MODE_OUTPUT);
    }

    // blink individual LEDs indefinitely
    for (auto pin : std::views::join(std::views::repeat(led_pins))) {
        gpio_set_level(pin, 1);
        std::this_thread::sleep_for(std::chrono::seconds{1});
        gpio_set_level(pin, 0);
    }
}
```

6 Assignment 2

Show that the ESP32 can turn three different LEDs on and off separately by sending commands over the serial interface.

6.1 Toolchain

Just like the previous one, for this assignment I have used the ESP-IDF toolchain. The ESP-IDF toolchain provides good C++ support. From the UART can be read using `std::cin`, but to get the usual behaviour, our program needs to initialize the UART driver first.

6.2 Software

```cpp
#include <iostream>
#include <map>
#include <ranges>

#include "driver/gpio.h"
#include "driver/uart.h"
#include "esp_vfs_dev.h"
#include "rom/gpio.h"

void toggle(gpio_num_t pin) {
    int level = gpio_get_level(pin) ^ 1;
    gpio_set_level(pin, level);
}

extern "C" void app_main() {
    // configure stdin to use blocking mode
    setvbuf(stdin, nullptr, _IONBF, 0);
    constexpr auto uart_num = CONFIG_ESP_CONSOLE_UART_NUM;
    uart_driver_install(static_cast<uart_port_t>(uart_num), 256, 0, 0, nullptr, 0);
    esp_vfs_dev_uart_use_driver(uart_num);

    const std::map<char, gpio_num_t> led_pins{
        {'0', GPIO_NUM_12},
        {'1', GPIO_NUM_14},
        {'2', GPIO_NUM_27},
    };

    // configure the pins for input and output
    for (auto pin : led_pins | std::views::values) {
        gpio_pad_select_gpio(pin);
        gpio_reset_pin(pin);
        gpio_set_direction(pin, GPIO_MODE_INPUT_OUTPUT);
    }

    for (char ch : std::views::istream<char>(std::cin)) {
        if (auto it = led_pins.find(ch); it != led_pins.end()) {
            toggle(it->second);
        }
    }
}
```

7 Assignment 4

Show that the ESP32 can turn three different LEDs on and off separately. Connect the ESP32 to a WiFi access point and host a webserver on the ESP32 to control the LEDs.

7.1 Toolchain

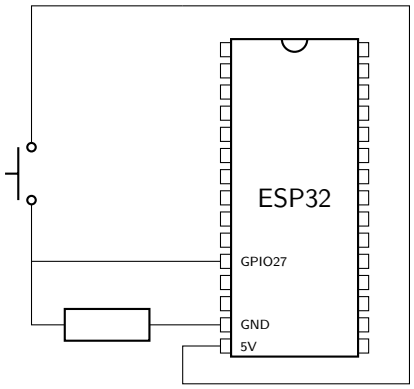
7.2 Software

`https://github.com/jochemarends/project-integration/tree/main/leds-wifi`

8 Final Assignment

For the final assignment, I am managing a database using SQL and the ESP32. I chose for this assignment because of the likelihood that the ESP32 has to interact with a database for the final product. The goal of this assignment is that the ESP32 will be able to log certain events, which will be stored into a database. The only event that will be logged is the pressing of a push button. The ESP32 will not communicate directly with the database; instead, it communicates with the database via a web server.

9 Demonstration Setup



I used the circuit above to connect the push button to the ESP32. The locations of the pins may differ depending on which version of the ESP32 is being used. The web server and the database are hosted a Raspberry Pi. I configured port forwarding so that the ESP32 can communicate with the web server from other networks.

9.1 Used Software Libraries

For the client, I have used libraries that come with ESP-IDF. These libraries allowed me among other things to establish a WiFi connection and perform Hypertext Transfer Protocol (HTTP) requests. For the server, I have used Go's standard library packages working with HTTP and SQL. I have also used a package that enables MySQL support for Go's database/sql standard library package. a Go driver for MySQL, which can be found at: `https://github.com/go-sql-driver/mysql`. I made use of Go's standard library for working with HTTP.

9.2 Software Architecture

The software consists of three major components: a HTTP client, a HTTP server, and a relational database. The ESP32 is communicating with the database via a web

server. The client makes HTTP requests to the server, after which server will query the database based on the request being made. Of these components, the server and the database are hosted a Raspberry Pi, while the client runs on an ESP32.

The HTTP client communicates with the HTTP server, which in turn communicates with the database. The ESP32 is interacting with the database via a web server as hosting a SQL client would be a very intensive task for a microcontroller.

9.3 Results

I did not experience any problems when working on this assignment. However, I did come accross some challenges for the design. At first instance I thought A SQL client is

9.4 Accountability

I have done this assignment by myself.

References

Espressif. (n.d.). *About Espressif.* https://www.espressif.com/en/company/about-espressif.

Espressif. (2024). *ESP32 Series Datasheet.* https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.