FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Wireless smart cube for time tracking with application Clockify
Student:	David Sobíšek
Supervisor:	Ing. Pavel Kubalík, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer engineering
Department:	Department of Digital Design
Validity:	until the end of summer semester 2024/2025

Instructions

1) Research existing solutions for wireless smart objects for easier time-tracking.
2) Design your solution based on the ESP8266 platform.
3) The designed solution will meet these requirements:
  - It will be tracking a project assigned by the user to a side tilted upwards.
  - The user will be able to communicate with the device using WiFi.
  - The tracked data will be sent to the application Clockify using their public REST API interface or stored on a microSD card when WiFi is not reachable.
  - LEDs inside the device will light up to indicate the actions and states of the device.
  - Its batteries will be charged wirelessly.
4) Implement your proposed solution and adequately test it.

Electronically approved by prof. Ing. Hana Kubátová, CSc. on 30 January 2024 in Prague.

Bachelor's thesis

WIRELESS SMART CUBE FOR TIME TRACKING WITH APPLICATION CLOCKIFY

David Sobíšek

Faculty of Information Technology
Department of Digital Design
Supervisor: Ing. Pavel Kubalík, Ph.D.
May 16, 2024

Contents

List of Figures

List of Tables

I want to thank my supervisor, Ing. Pavel Kubalík, Ph.D., for his time, effort and guidance. My gratitude goes to my family, close friends and colleagues for their consistent support throughout my academic journey.

Declaration

In Prague on May 16, 2024

Abstract

The bachelor thesis focuses on designing and implementing a smart cube for time tracking using the application Clockify. The physical cube serves as a visual aid for the user to start, switch, or terminate time tracking for projects set in the Clockify web application and assigned to each side of the cube. After researching existing solutions on the internet, a custom implementation was designed based on the ESP8266 development platform combined with the GY-521 (MPU6050) gyroscope and accelerometer module. The device is completely wireless thanks to the support of wireless charging and can work for a limited time without a WiFi connection. In that case, the project records are stored on a micro SD card and sent to the web application when the cube is reconnected to WiFi.

Keywords time-tracking smart cube, visual aid for employees, web application Clockify, ESP8266, GY-521 with MPU6050, wireless charging, WiFi, micro SD card

Abstrakt

Bakalářská práce se zaměřuje na návrh a implementaci chytré kostky pro sledování času s aplikací Clockify. Fyzická kostka slouží jako vizuální pomůcka pro uživatele, který díky ní spouští, přepíná nebo vypíná sledování času u projektů, které má uložené ve webové aplikaci Clockify a které přiřadil k jednotlivým stranám kostky. Po prozkoumání existujících řešení na internetu byla navržena vlastní implementace založená na vývojové platformě ESP8266 v kombinaci s modulem gyroskopu a akcelerometru GY-521 (MPU6050). Zařízení je kompletně bezdrátové díky podpoře bezdrátového nabíjení a je schopné pracovat omezený čas bez připojení k WiFi, kdy se záznamy o jednotlivých projektech ukládají na micro SD kartu a do webové aplikace se odešlou po připojení kostky k WiFi.

Klíčová slova chytrá kostka pro sledování času, vizuální pomůcka pro zaměstnance, webová aplikace Clockify, ESP8266, GY-521 s MPU6050, bezdrátové nabíjení, WiFi, micro SD karta

List of abbreviations

ACK	Acknowledge
API	Application Programming Interface
CC/CV	Constant-Current/Constant-Voltage
CPU	Central Processing Unit
CLI	Command Line Interface
CS	Chip Select
EEPROM	Electrically Erasable Programmable Read-Only Memory
FAT	File Allocation Table
FS	File System
GND	Ground
GPIO	General Purpose Input/Output
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
I^2C	Inter-Integrated Circuit
I/O	Input/Output
IoT	Internet of Things
LE	Low Energy
LED	Light-Emitting Diode
LiPol	Lithium Polymer
MCU	Microcontroller Unit
MISO	Master Input/Slave Output
MOSI	Master Output/SlaveInput
PCB	Protection Circuit Board
PWM	Pulse Width Modulation
RFID	Radio Frequency Identification
RGB	Red, Green, Blue
SCL	Serial Clock
SCLK	Serial Clock
SD	Secure Digital
SDA	Serial Data
SoC	State of Charge
SPI	Serial Peripheral Interface
SS/CS	Slave Select/Chip Select
SSID	Service Set Identifier
TWI	Two Wire Interface
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

Chapter 1
Introduction

In today's fast-paced work environments, professionals paid by the hour often face challenges in accurately monitoring their time across diverse projects and tasks. Amid juggling multiple responsibilities, it's common to overlook initiating the time-tracking software or switching tasks correctly while multitasking throughout the day. These issues can lead to billing inaccuracies, project management inefficiencies, and difficulties in maintaining comprehensive work records.

While existing software solutions attempt to tackle these challenges, physical time-tracking tools offer distinct advantages to overcome them by leveraging visual and muscle memory. Unlike software interfaces, physical tools provide tangible reminders and intuitive interactions that engage users on a tactile level.

However, existing physical time-tracking devices have notable limitations. Many require users to adopt unfamiliar proprietary software, disrupting established workflows and potentially causing frustration due to lacking unique features found in preferred applications. Moreover, maintenance and support for these devices can be costly, leading to a user experience spoiled by subscription fees, software obsolescence, or loss of support over time, rendering the device obsolete.

This thesis endeavours to address these issues by proposing the design and future implementation of a time-tracking device that integrates seamlessly with Clockify, an activity-tracking platform widely used by professionals. The envisioned solution seeks to overcome the identified problems and offer an alternative to existing activity-tracking products. Leveraging Clockify's popularity in office environments, this device intends to simplify its integration into daily work life. Additionally, the thesis explores the potential to support other tracking applications, leveraging the flexibility of an underlying backend architecture.

The proposed solution incorporates light signals as visual reminders for ongoing activity tracking. When users become accustomed to seeing a lit-up cube while working, they should quickly notice when they forget to activate tracking, reinforcing consistent time management practices. Similarly, the cube's consistent colour after switching activities serves as an intuitive cue to support task-switching and project management, ultimately aiming to enhance efficiency and accuracy in time tracking.

Chapter 2

Objectives

The primary aim of this thesis is to design, develop, and test a functional prototype of a wireless smart cube designed to simplify activity tracking using the Clockify API. This device will be based on the ESP8266 platform, integrating a GY-521 gyroscope module. Its primary functionality will involve initiating and terminating task time tracking in Clockify based on the cube's tilt. Users will assign tasks to the cube's sides in a form shown to them after connecting to a WiFi access point created by the cube. The currently tracked project will be indicated by illuminating the cube with the corresponding colour. The smart cube will be configured via WiFi and will support wireless charging for added convenience.

The "Existing Solutions" chapter will research and analyze the existing physical time-tracking devices, assessing their strengths and weaknesses. The results will guide the component selection outlined in the following "Analysis" chapter.

Subsequent chapters including "Suggestion of the Solution," "Implementation of the Hardware," and "Implementation of the Program" will detail the assembly of suggested components and the necessary software development to realize the smart cube prototype. These sections will delve into the control program's and hardware design intricacies, highlighting the libraries used for various functionalities and the implemented features.

Lastly, the "Testing" chapter will describe and evaluate the prototype's testing process, evaluating its functionality and accuracy.

Chapter 3
Existing Solutions

The objective of this chapter is to analyze existing activity-tracking devices available for online purchase. Three devices were selected for comparison, each demonstrating both distinct and similar advantages and disadvantages, including variations in shape, functionality, and user experience.

3.1 TIMEFLIP2

The twelve-sided tracker TIMEFLIP2 lets users track time spent on projects via Bluetooth 4.0 LE using the company's web, iOS, or Android app. The device runs on 2 alkaline AA batteries with a claimed average of 40 hours of tracking. It works offline, too, and tracks up to 1066 flips without connection. The device weighs 90 grams, and it supports tap-on to start or pause activities, signalling to the user with a blink from the embedded LED light shining through translucent walls, which project they are currently tracking. [1]

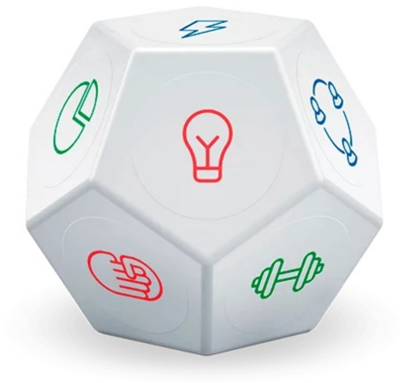


Figure 3.1 TIMEFLIP2 [1]

Relying on alkaline batteries presents challenges due to maintenance requirements and potential leakage. To address this, the author decided to use a more modern LiPol battery. This will also let the proposed solution offer a convenient added feature, thanks to the compatibility of the wireless charging module with the battery.

The second drawback the author has found with TIMEFLIP2 is that it provides a custom software solution for the user. One of the goals for the proposed device is so that the user is able to seamlessly add the device into their workflow without having to transition to new software and sacrifice some of their favourite features unique to that software.

The author took inspiration from the way this device communicates with the user. The lights inside the tracker can be seen as a visual cue for the user to remember to start tracking and can be used as a means of all the communication needed as well. However, additional sounds or haptics could be disruptive in the office environment for the user's colleagues, so only light signals will be used.

3.2 Timeular Tracker

Timeular Tracker has eight sides but claims to let the user track 1000 activities. That is possible because it connects to a desktop app and one side of the device can be used as a shortcut to open a menu with various activities they can preset. The app is available for all major platforms, such as Windows, Linux, and macOS, but it is possible to use the tracker without the menu shortcut with a web, Android, or iOS app too. There is not much information about the physical device on the website, only that it weighs 85 grams, should last 6 months per charge, and is charged using the included dock. There is a single small LED on the edge of the device, but the purpose is not mentioned on the website, it only seems to serve as an indicator to show if the device is turned on or if the battery is low, not to show the selected project. [2]

The shape of the device seems less practical than a regular cube since it is not balanced, and it can charge only standing upright with the use of the included dock, which doesn't seem very stable from the photos. Also, since it doesn't light up in the colour of the project, the user has to label the sides with a marker or stickers to recognize which side task is being tracked. This goes against the goal of triggering the user's visual memory with the cube not being lit up when they arrive at work.

The biggest advantage comes with the use of proprietary software and the preset menu action, but that requires the user to transfer from currently used software, which brings the same problem as explained with the TIMEFLIP2 device.

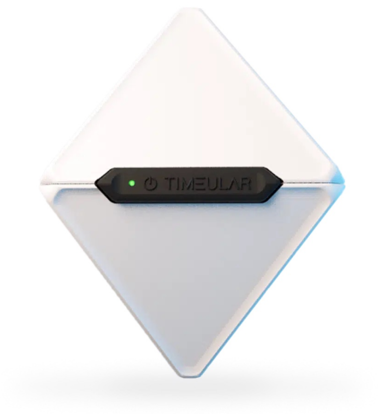


Figure 3.2 Timeular Tracker [2]

One of the most compelling features of this device is its battery life. The Timeular Tracker can remain active for extended periods due to its utilization of Bluetooth LE (Low Energy), which is significantly more power-efficient compared to WiFi. Additionally, the absence of LED lighting contributes to its longevity.

In contrast, the proposed solution may exhibit a relatively shorter battery life because it relies on WiFi for communication with the tracking application's API. However, if Bluetooth LE were adopted, similar to the Timeular Tracker, it would enhance power efficiency but require the use of a smartphone or laptop to transmit data to the tracking application.

3.3 Adafruit IO Time Tracking Cube

The Adafruit IO Time Tracking Cube is not a commercially available product but rather an instructional guide utilizing Adafruit products. To assemble this project, users are instructed to buy an Adafruit Feather HUZZAH, an LED light strip, and encase them within a 3D-printed housing. This setup enables users to track time and recommends using Microsoft Excel tables for data management. The cube provides audio-visual cues by illuminating its translucent walls with LED lights corresponding to the currently tracked project and emitting a sound via a piezo buzzer upon recognizing tilt movements. [3]

The use of a piezo buzzer for tracking notifications presents a disruptive drawback, as its audible alerts could disturb others sharing the same office space. Another limitation is relying on Excel for tracking, which lacks automation, real-time capabilities and any possibility to add unique features to the tracking experience.

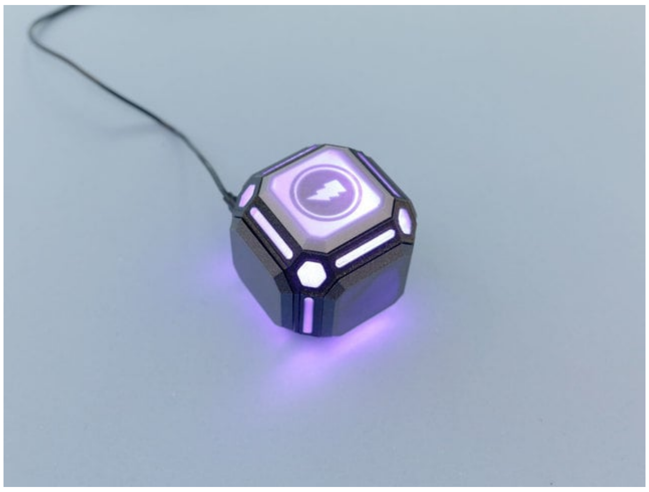


Figure 3.3 Adafruit IO Time Tracking Cube [3]

This project revolves around leveraging an ESP8266-based Adafruit platform for WiFi communication with tracking software, which inspired the adoption of a similar ESP8266-based platform in this thesis design. The cube's multi-sided design is ideal for accommodating various functions and project-tracking capabilities within the proposed solution. Drawing inspiration from this design, it serves as a foundational reference for developing the smart cube prototype, incorporating valuable insights to enhance its design and functionality.

3.4 Summary

The Table 3.1 shows the summary of the advantages and disadvantages of each of the devices:

After looking at the disadvantages in the table, it becomes evident that one common drawback across these devices is their reliance on proprietary software, which limits user flexibility and integration with existing workflows. Given this common challenge, this thesis will focus on addressing this limitation by developing a smart cube prototype that seamlessly integrates with a widely used productivity application without requiring users to transition to proprietary software. By prioritizing open-source and adaptable design principles, this thesis aims to provide a user-friendly solution that enhances user experience and productivity in various office environments.

Table 3.1 Summary of Existing Time Tracking Devices

Device	Advantages	Disadvantages
TIMEFLIP2	- Bluetooth connectivity for seamless integration with web and mobile apps - offline tracking capability with a large flip capacity - user-friendly tap-on feature to start or pause activities	- reliance on alkaline AA batteries - custom software solution may not integrate well with existing workflows
Timeular Tracker	- integration with desktop and mobile apps for versatile usage - long battery life using Bluetooth LE - proprietary software with preset menu actions for quick activity selection	- imbalanced shape and docking requirement for charging - requires a transition to proprietary software - lack of visual project indication without additional labelling
Adafruit IO Time Tracking Cube	- DIY project using accessible components - offers audio-visual cues for project tracking - utilizes open-source instructions for customization	- requires assembly and customization - relies on manual data management using Excel - user possibly has to switch from tracking software to Excel - buzzer sounds could be disturbing in offices

Chapter 4

Analysis

This chapter delves into the selection and analysis of key components, modules, and communication methods for the implementation of the time-tracking device. The options are chosen to fit the device's purpose best while maintaining the low cost.

4.1 The Control Unit

The control unit functions as the central processing unit of the device, overseeing all its operational tasks. Its primary responsibilities include time tracking for various tasks and communicating the collected data to designated tracking software. While the control unit handles the tracking logic for offline and online functionality of the device, other duties are streamlined and managed by the server, as detailed in subsection 5.2.3. This setup offers the distinct advantage of flexibility, offering to allow users the switch between different tracking software options seamlessly without requiring firmware updates or disassembly. To leverage server capabilities, the device must connect to WiFi, emphasizing the advantage of utilizing a development board equipped with integrated WiFi capabilities.

Initially, various Arduino boards were considered for this project; however, mainly due to cost considerations, the decision was made to employ an ESP8266-based development kit. The ESP8266 platform, developed by Espressif Systems along with its successor ESP32, offers a cost-effective solution with built-in WiFi and Bluetooth capabilities, making it highly suitable for IoT applications [4]. Utilizing bare ESP32 or ESP8266 chips directly is impractical for most developers, as it lacks essential circuitry for power management, computer connectivity, code uploading, and peripheral integration. Development boards based on these platforms provide a user-friendly interface, simplifying the development process by offering essential features and connectivity options via accessible pins. [5]

4.1.1 Wemos D1 R2

The Wemos D1 R2 development board is built around the affordable and highly capable ESP8266 MCU (microcontroller unit) developed by Espressif Systems. Featuring a micro USB connector for programming and Arduino compatibility, the board offers seamless integration with existing development environments. One of its standout features is its built-in WiFi module, providing wireless connectivity essential for IoT applications. It has 11 digital inputs/outputs, 10 channels of PWM, 1 analogue input, as well as popular communication interfaces, all working on a voltage of 3.3V. The module also provides a 3.3V output pin that can be used to charge other devices. [6].

The decision to use the Wemos D1 R2 was primarily driven by its low cost and integrated WiFi capabilities, aligning perfectly with the project's requirements. Additionally, the board's pinout closely resembles that of the popular Arduino Uno, facilitating ease of use and compatibility with a wide range of existing Arduino-based projects and peripherals. This familiarity streamlines the development process, enabling rapid prototyping and efficient integration of the ESP8266 platform into the time-tracking device. The combination of affordability, wireless connectivity, and Arduino compatibility makes the Wemos D1 R2 an ideal choice for this project, enhancing both functionality and development flexibility.

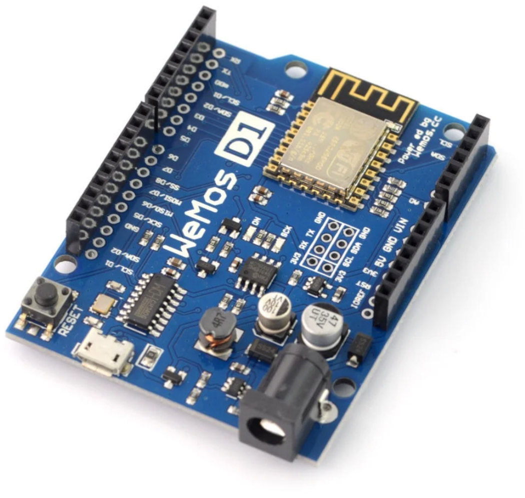


Figure 4.1 Wemos D1 R2 [6]

4.1.1.1 MCU Pinout

The board provides a total of 11 usable GPIO (General Purpose Input/Output) pins; however, not all of these pins can be used without certain limitations. GPIO0 (corresponding to D3 on the Wemos D1 R2) is pulled up by a 10kΩ resistor and is connected to the FLASH button. During boot, pulling GPIO0 LOW can cause the boot process to fail. Similarly, GPIO2 (D4) is pulled up by a 10kΩ resistor and is linked to the on-board LED, which is active when the pin is LOW. These pins are typically used for I^2C communication, alongside D14 and D15.

GPIO1 and GPIO3 (D0 and D1) serve as Rx and Tx connections for serial communication. If serial communication is not utilized, they can function as general GPIO pins. It's crucial to set these pins to HIGH during boot, as pulling GPIO1 LOW can disrupt the boot process.

GPIO15 (D8) is equipped with a fixed 10kΩ pull-down resistor, and pulling this pin HIGH during boot can lead to boot failure.

GPIO16 (D0) is specifically used to wake the ESP8266 from deep sleep mode. [7] [8]

4.1.1.2 Deep Sleep Problematic

The ESP8266 microcontroller offers three sleep options to conserve battery life when running on a limited power source, such as a battery. Sleep modes deactivate power-intensive activities like WiFi and the system clock to reduce power consumption. [9]

Among the available sleep modes on ESP8266, deep sleep mode is the most power-efficient. In this mode, WiFi, the system clock, and the CPU are powered down, leaving only the RTC operational. This results in an average of approximately 1000 times lower current consumption than in modem sleep mode, where only WiFi is disabled. However, achieving consumption this low is impossible with full-featured development boards like the Wemos D1 R2. [9]

Deep sleep mode is ideally suited for applications where the microcontroller needs to wake up periodically to perform a specific task and can remain in sleep mode for predefined intervals between operations. [9] However, deep sleep mode can be triggered for an unspecified period, and the wake-up signal can come from a reset or an external button. This capability can be leveraged in the proposed device during the charging process, enabling the microcontroller to remain inactive while being charged until it is woken up to resume tracking.

Furthermore, this feature holds potential for future extensions of the solution. For example, powering the gyroscope and accelerometer modules from a separate power source would allow these components to remain operational even when the MCU enters deep sleep mode. This setup could enable the device to remain in a low-power state while continuously monitoring orientation, waking up only briefly to track activity upon flipping the cube.

4.2 Orientation In Space

The control unit requires additional components to enable orientation detection in space, specifically an accelerometer and gyroscope module. After considering various options, the GY-521 with the MPU6050 module was selected based on the author's familiarity and positive experience with its performance.

4.2.1 GY-521 with MPU6050 Gyroscope and Accelerometer Module

The GY-521 is a three-axis gyroscope and a three-axis accelerometer MPU6050 module operating on 3.3V, making it well-suited for the selected control unit. This module excels in detecting both static gravitational effects and dynamic sensor movements, providing comprehensive orientation-sensing capabilities crucial for the time-tracking device's functionality. Additionally, the module communicates using the I^{2}C (Inter-Integrated Circuit) standard, ensuring efficient and reliable data transmission between the module and the control unit. [10]

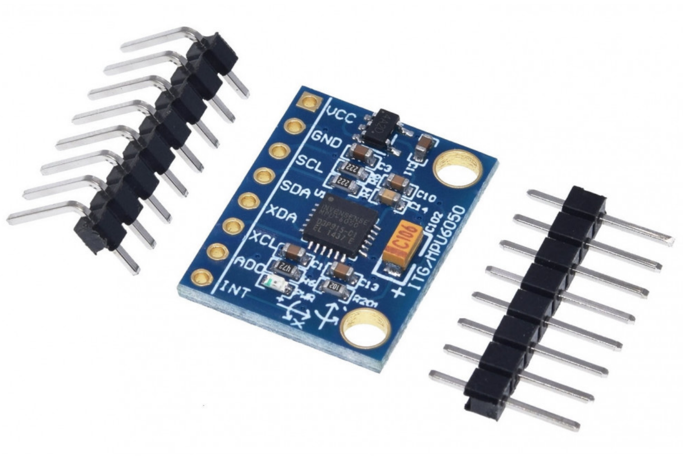


Figure 4.2 GY-521 with MPU6050 Gyroscope and Accelerometer Module [10]

By incorporating the GY-521 module into the device design, the control unit gains the ability to accurately detect orientation changes and dynamic movements, essential for precise time-tracking based on the cube's orientation. This module's compatibility with the control unit's voltage levels and communication standards ensures seamless integration and reliable performance within the overall system.

4.3 Charging

Given that the device will primarily be situated on an office desk, which is often already filled with a multitude of cables, especially in technology-oriented environments, the integration of wireless charging is a logical choice. This approach eliminates the need for additional cables cluttering the workspace and avoids the inconvenience of a permanent cable protruding from the device.

4.3.1 Batteries

To ensure the device maintains uninterrupted operation throughout a full workday without requiring charging, a rechargeable LiPol battery 104050 with a capacity of 2500mAh and a nominal voltage of 3.7V will be integrated. This battery configuration consists of a single prismatic cell in a 1-series, 1-parallel setup, offering sufficient capacity for extended use. [11]

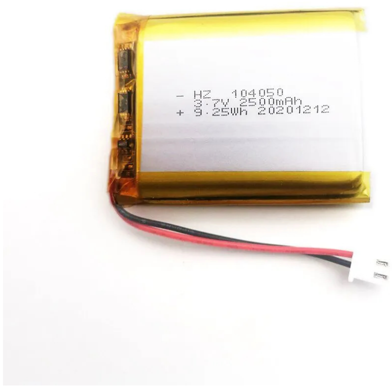


Figure 4.3 Used Battery [11]

Despite the presence of an integrated battery PCB (protection circuit board) designed to prevent issues such as over-charge, over-discharge, over-current, and short-circuiting, an additional protective charging module will be implemented for enhanced safety and longevity. [11] This supplemental measure ensures the reliable and safe operation of the battery within the device, mitigating potential risks associated with battery management.

4.3.2 Wireless Charging Module

To enhance user convenience and maintain a tidy workspace, the device will utilize a wireless charging module. The selected module is designed for various electronic devices, offering a 5V output at 1A. Wireless charging operates effectively through non-conductive materials such as wood and plastic, ensuring seamless integration into the device's design and functionality. [12]

By integrating wireless charging technology, the device remains unobtrusive and user-friendly, facilitating effortless charging without the hassle of traditional cables. This feature aligns with the device's goal of providing a streamlined and efficient solution for time-tracking in office environments.

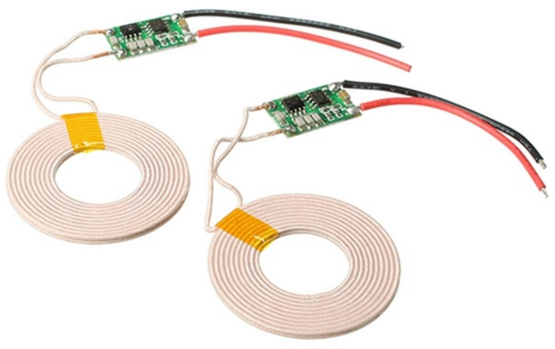


Figure 4.4 Wireless Charging Module [12]

4.3.2.1 Testing

Prior to integration into the final design, the module was tested by the author. The testing confirmed that for effective charging, the coils of the module need to overlap with at least half of their area and remain less than 5mm apart. During charging, significant energy loss occurs, which worsens with increased distance and decreased coil overlap. Therefore, it is advisable to incorporate a ridge into the design of the cube's casing to ensure optimal alignment with the charging pad. Nevertheless, the module has demonstrated sufficient performance for the prototype, thus there should be no issues in its utilization.

4.3.3 TP4056 Charging Module

For safe battery charging, a TP4056 USB-C charging module will be used, utilizing the CC/CV (constant-current/constant-voltage) charging method to ensure efficient and safe charging of the batteries. [13] This charging approach involves two phases: initially delivering a constant current for rapid charging, followed by a transition to a constant voltage maintenance mode to slow down charging as the battery reaches its capacity. [14]

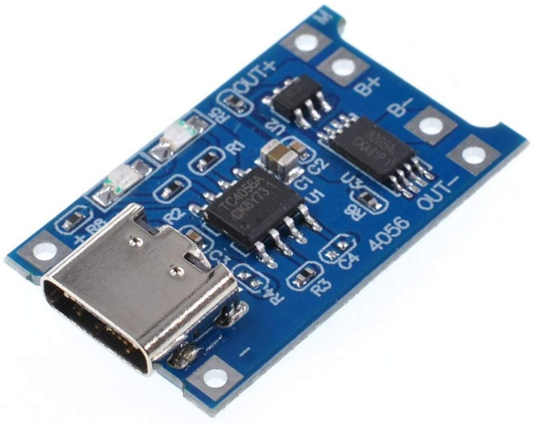


Figure 4.5 TP4056 Charging Module [13]

The TP4056 module can be powered either via a USB-C cable or through direct connections to the + and - pins. It necessitates a power source capable of delivering at least 1A to effectively charge the battery. [13] This requirement is well met by the wireless charger module used, which provides a sufficient 1A output.

One crucial consideration with the TP4056 module is its lack of support for ”power-load-sharing,” necessitating that any load be disconnected while the battery is being charged. The reason for this is that the charging circuitry detects when the charge current falls below 1/10 of the matter capacity (constant current charge mode near the end of the charge cycle). When a load is connected to the battery, this changes the detected current, allowing the TP4056 to continue charging, which may never end and damage the battery. [15] However, given the integrated protection of the battery against over-charging and the device being in deep sleep mode during charging (thanks to the placement of the charger and the orientation of the device while utilizing it), where the power consumption is much lower, this limitation should not pose an issue for the device's operation and battery health.

An area of future work that extends beyond the scope of this thesis involves implementing a mechanism to detect the charging status of the device and subsequently enabling an automatic shutdown for enhanced safety. This feature aims to enhance the overall safety and reliability of the device during charging cycles, ensuring the user cannot damage the device.

4.3.4 LaskaKit Fuel Gauge MAX17048

Since the device will not be able to charge while using, to monitor battery status and ensure timely user notification of low battery conditions, the LaskaKit Fuel Gauge MAX17048 module will be integrated into the device.

This lithium-polymer battery meter communicates via the I^2C bus, enabling measurement of battery voltage and state of charge (SoC) using the ModelGauge proprietary algorithm. The module can generate interrupts based on battery conditions such as low state of charge, under-voltage, or over-voltage. [16] By leveraging these features, the device can effectively alert users to critical battery levels, ensuring uninterrupted usage and timely recharging when needed.

4.3.5 Step-up Voltage Regulator

Step-up voltage regulators raise the input voltage to a higher output level. They temporarily store energy and release it at a higher voltage. It is vital to use them in battery-powered devices to provide stable voltage to the microcontroller. [17]

The SX1308 step-up converter is selected for this purpose due to its efficiency and output voltage range. With an efficiency rating of 95% and the ability to output voltages between 3V to 28V, the SX1308 is an ideal choice to provide a stable 5V supply to the microcontroller circuit. This efficient conversion ensures optimal performance and reliability in the device's power management system. [18]

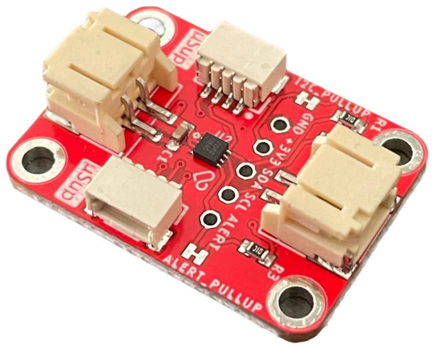


Figure 4.6 LaskaKit Fuel Gauge MAX17048 [16]

4.4 Memory

The ESP8266 microcontroller emulates Arduino's EEPROM functionality using a designated area within its Flash memory. This nonvolatile memory retains data even when the board is powered off, offering persistent storage. However, it comes with limitations such as a finite size and lifespan. While the Flash memory on ESP8266 boards typically boasts a capacity of 4MB, the emulated EEPROM segment is limited to 4kB and has a maximum write endurance of 100,000 cycles. [19] Given the potential high frequency of write operations in our time-tracking cube, relying solely on EEPROM is impractical and may lead to premature wear.

4.4.1 Micro SD Card Module

To address this challenge, a micro SD card module will be incorporated into the design. This module serves as a reliable backup offline storage solution, enabling the cube to store tracking data locally in situations where a WiFi connection is unavailable. The micro SD card communicates with the control unit via SPI (Serial Peripheral Interface), a widely used protocol for data exchange between microcontrollers and peripheral devices. Operating at 3.3V, the micro SD card module is compatible with the ESP8266 microcontroller, ensuring seamless integration within the smart cube system. [20]

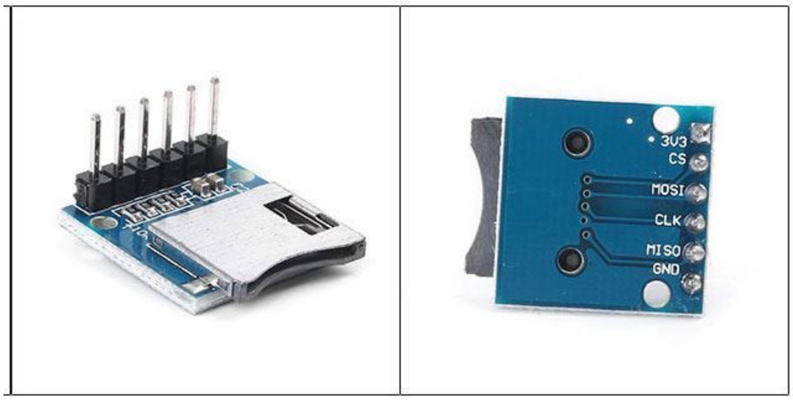


Figure 4.7 Micro SD Card Module [20]

4.5 Real-Time Clock

Given the critical requirement of accurate timekeeping for the device, integrating a dedicated RTC (real-time clock) module is highly advantageous. While the ESP8266 includes an onboard RTC, it primarily functions as a timer and is prone to significant time drift over extended periods of operation. The internal RTC's accuracy may not suffice for precise timekeeping needs.

Obtaining the current time and date from network synchronization on each power-up is a viable option; however, this approach necessitates an internet connection for each time initialization. This dependency poses a challenge, as the device would be unable to track time accurately during the initialization phase without internet access.

To address this issue effectively, an external RTC module is employed. These RTC modules are purpose-built for accurate timekeeping, offering reliable clock functions independent of the internet or device power cycles. By integrating an RTC module with the ESP8266, the device ensures consistent and precise timekeeping, crucial for its operational reliability and functionality.

4.5.1 Waveshare PCF8563

The Waveshare PCF8563 real-time clock (RTC) module is designed for optimal low power consumption, making it an ideal choice for energy-efficient applications. It communicates via the I^2C bus at a maximum speed of 400 kbit/s. The timekeeping accuracy of the PCF8563 is derived from a high-precision 32.768 kHz quartz crystal oscillator, providing stable and reliable clock signals. This oscillator serves as the foundation for tracking essential time parameters, including year, month, day, weekday, hours, minutes, and seconds. [21]

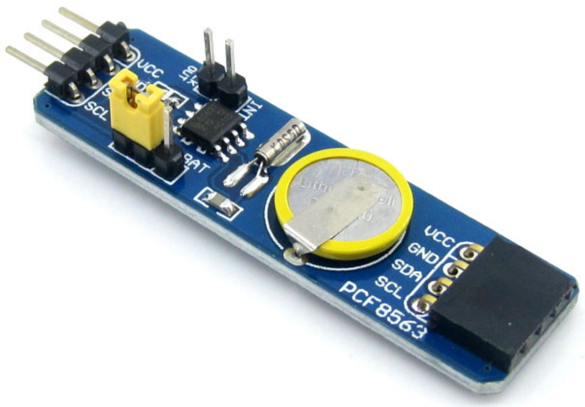


Figure 4.8 PCG8563 RTC Module [21]

One of the standout features of the PCF8563 module is its ability to maintain accurate time even during power-down periods. This is made possible by an onboard 3.3V battery that powers the RTC independently of the main device. As a result, the module retains the correct time and date information, ensuring continuous operation without relying on external power sources. [21]

Additionally, the PCF8563's calendar system includes support for leap years, ensuring accurate date calculations even in leap year scenarios. This feature enhances the module's versatility and compatibility with a wide range of time-sensitive applications. [21]

4.6 Communication

To enable communication between the various modules and the control unit within the time-tracking cube, multiple protocols will be employed. Each protocol serves specific purposes and offers unique advantages for efficient data exchange and control. The protocols will be UART, SPI and I^{2}C.

4.6.1 SPI

The SPI, or Serial Peripheral Interface, protocol is very common among various devices; for example, SD card reader modules, RFID card reader modules, and 2.4GHz wireless transmitters/receivers all use it. Its unique benefit is that the data is sent continuously in a stream without interruptions. [22]

The devices using SPI for communication are in a master-slave relationship, where the master is the controlling device (usually the microcontroller) and the slave takes instructions from it. The most basic system is one master and one slave, but multiple slaves can be controlled by one master too. [22]

There are four lines connecting the master and the slave. MOSI, or Master Output/Slave Input, is the line for the master to send the data to the slave, MISO, or Master Input/Slave Output, on the other hand, is for the slave to send the data to the master. SCLK is the clock signal line and finally, SS/CS is the Slave Select/Chip Select line to choose which slave to send the data to. [22]

In the implementation, SPI will be needed for the communication between the microcontroller and the micro SD card module.

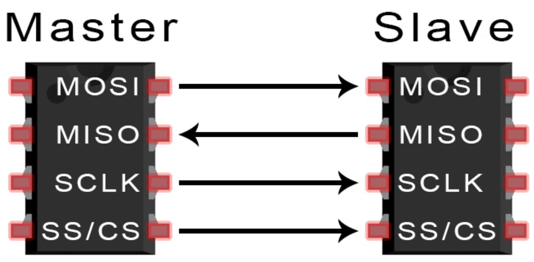


Figure 4.9 SPI [22]

4.6.2 UART

UART stands for Universal Asynchronous Receiver/Transmitter. It is not a protocol like SPI or I^{2}C but a physical circuit in a microcontroller instead to transmit and receive serial data. It only uses two wires for communication. [23]

The transmitting UART communicates directly with the other UART. It converts parallel data from a controlling device into serial form and transmits it to the receiving UART, which then converts the serial data back to parallel for the receiving device. Data flows from the Tx pin to the other UART's Rx pin. The data is transmitted asynchronously, which means there is no clock signal, and it is sent in packets bordered by the starting and the ending bits. The incoming bits are read at a frequency known as the baud rate, a measure of the speed of data transfer, expressed in bits per second. Both of the UARTs have to operate at almost the same baud rate, it can differ by about 10%. [23]

UART will be used in the communication between the microcontroller and the computer for testing purposes and a baud rate of 115200 bps will be used.

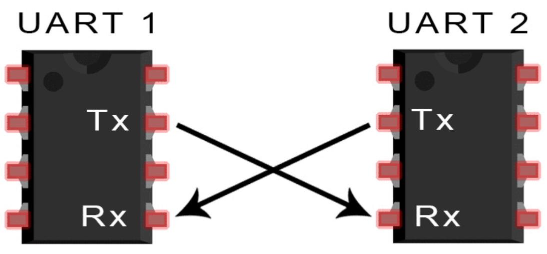


Figure 4.10 UART [23]

4.6.3 I^{2}C

I^{2}C is a protocol that combines the best features of SPI and UART. You can connect multiple slaves to a single master like SPI and you can also have one or multiple slaves controlled by multiple masters. It only uses two wires to communicate between devices, such as UART. The lines are called SDA (or serial data) for sending and receiving data and SCL (or serial clock), which is the line that carries the clock signal. The data is transferred bit by bit along a single wire (SDA). The master always controls the SCL signal and thanks to the clock signal being shared between the master and the slave, I^{2}C is synchronous. [24]

The data is transferred in messages containing an address frame (7 to 10 bits) because I^{2}C doesn't have a slave select line like SPI. This address frame is sent to all the slaves connected to the master, then it is compared to the address that the slaves address and if it matches, the slave sends an ACK (acknowledge) bit back to the master. The last bit of the address frame also includes information about whether the master wants to read the data from the slave or write it to it. The data frame is always 8 bits long. [24] The used I^{2}C addresses are shown in the Table 4.1.

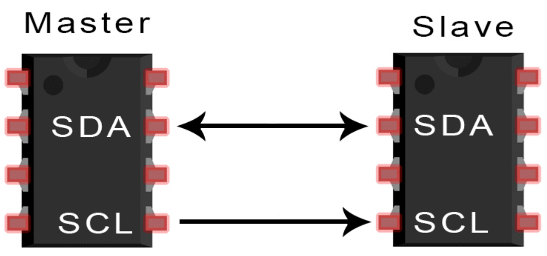


Figure 4.11 I^{2}C [24]

Table 4.1 I^{2}C Addresses

Component	Address
GY-521	0x68
PCF8563	0x51
MAX17048	0x36

4.7 Programming and Testing

For programming and testing the device, the micro USB port on the MCU serves as the primary interface. This port enables direct communication between the development environments (Arduino IDE and PlatformIO) and the MCU using a UART serial connection.

4.7.1 Arduino IDE

The Arduino IDE is a popular choice for programming microcontrollers, including the ESP8266. To use Arduino IDE with the ESP8266, the IDE has to be configured appropriately. Once set up, writing, compiling, and uploading code to the ESP8266 through the Arduino IDE interface is possible. The IDE includes a Serial Monitor feature that allows to monitor the device's output and debug messages over the UART serial bus. Arduino IDE is user-friendly and widely used, making it suitable for beginners and experienced developers alike. [25]

4.7.2 PlatformIO

PlatformIO is an open-source plugin built on top of Microsoft Visual Studio Code, intended for IoT development, supporting multiple platforms, including the ESP8266. It offers a more advanced and flexible development environment compared to Arduino IDE. With PlatformIO, it is possible to manage libraries, dependencies, and project configurations more efficiently. The integration within Visual Studio Code provides a seamless development experience with features such as IntelliSense, debugging, and project management. [26]

Chapter 5

Suggestion of the Solution

This chapter proposes the design of the program and the backend, along with the module connections. The suggested solution should meet the thesis goals, such as user configuration via WiFi, accurate Clockify app integration or saving the tracking data on the micro SD card.

5.1 Suggestion of the Modules Connection

The control unit is connected directly to most modules, as depicted in the Figure 5.1.

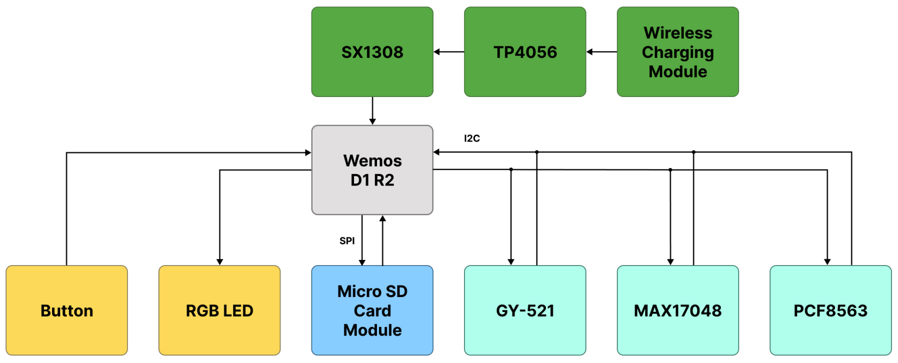


Figure 5.1 Block Schematic of the Hardware Modules Connection

The gyroscope and accelerometer, charging meter module, power button, and RGB LED must communicate with the control unit using I^2C. The micro SD card module communicates via the SPI bus. The charging module provides power to the batteries and from the batteries to the MCU through the stabilizing step-up regulator.

5.2 Program Design

This section suggests how the control program is developed and implemented and explains why a backend is needed.

5.2.1 Used IDEs

There are two major IDE options that can be used to write the control program for the ESP microcontroller.

The first one is Arduino IDE, an open-source software designed for programming Arduino boards. The IDE supports all the major operating systems (Windows, macOS and Linux). [25]

However, an add-on is necessary to enable programming for the ESP8266 microcontroller. A community-developed add-on brings the support for the ESP8266 chip to the Arduino environment and lets the users leverage all Arduino functions and libraries and run them directly on ESP8266. The package should be added as an Additional Board Manager in the Preferences of the IDE and then the ESP8266 platform should be installed. [27]

The second option is PlatformIO. It is an IDE solution for Microsoft Visual Studio Code intended for the creation and delivery of embedded products. It offers smart code completions based on variable types, function definitions and library dependencies. The users can use the built-in Terminal with PlatformIO Core (CLI) and a Serial Port Monitor. [26]

5.2.2 Control Program

The control program is structured into smaller files with the implementation of the individual classes to keep it clean and simple. Each will handle specific functionality and phase of the program lifecycle. The basic software design schematic can be seen in the Figure 5.2.

5.2.2.1 Program Phases

The user configuration phase is triggered after the initial press of the button on top of the cube or after the cube is turned to the opposite side dedicated to the user configuration. This phase sets up a WiFi access point for the user interaction. The user connects to it using a WiFi-capable device (e.g., smartphone, laptop) to access a simple configuration form on a webpage.

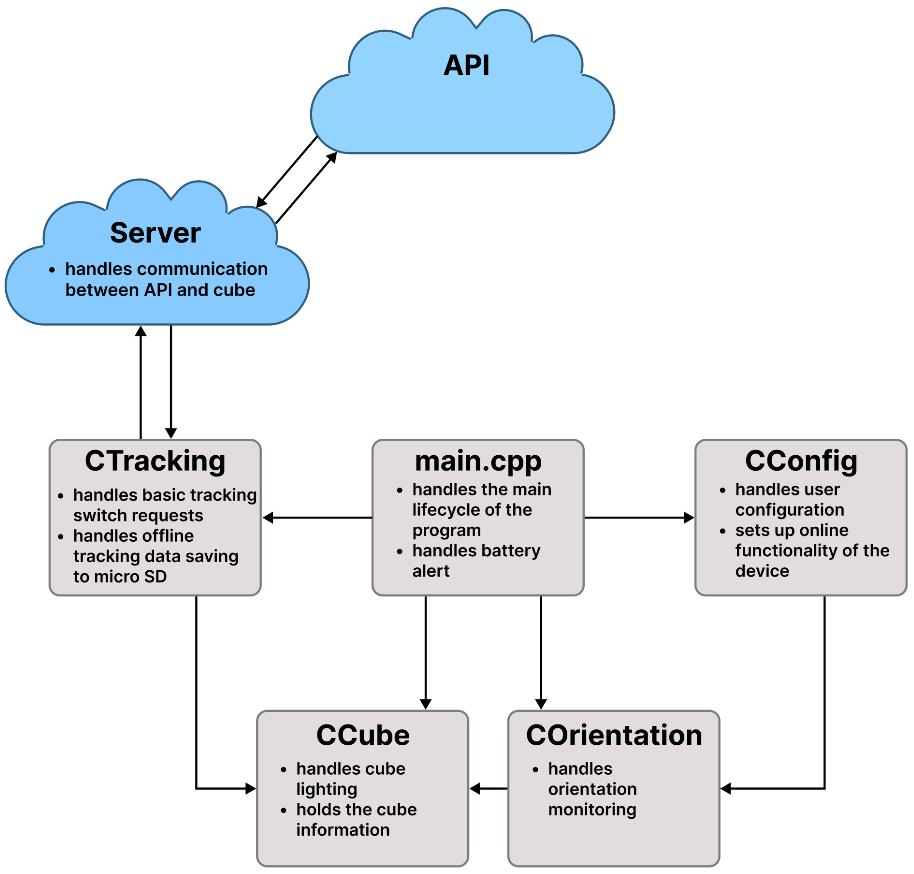

API
Server
• handles communication between API and cube
CTracking
• handles basic tracking switch requests
• handles offline tracking data saving to micro SD
main.cpp
• handles the main lifecycle of the program
• handles battery alert
CConfig
• handles user configuration
• sets up online functionality of the device
CCube
• handles cube lighting
• holds the cube information
COrientation
• handles orientation monitoring

Figure 5.2 Block Schematic of the Software Design

For the initial setup, the user completes the form, providing essential information, such as the personal API key generated in the Clockify application or the WiFi network name (SSID) and password for the control unit to connect to the internet. This configuration is then saved to the device memory for subsequent usage to gather data about the user's projects, which they can assign to the sides of the cube.

For the reconfiguration of the set values, another form is shown, where the projects can be assigned to the cube sides, the WiFi network can be changed to a new one and the device can be completely reset.

The tracking phase encompasses a key function: it manages task tracking by determining whether data should be sent to the web application or stored locally. This decision is crucial for ensuring flexibility and data security based on user preferences and connectivity status.

Additionally, the cube's power level is monitored while running. If the device is running low on battery, the cube signals it to the user.

5.2.3 Server design

For the suggested design to support various future tracking software options, a server is essential to handle simplified HTTP requests containing user information, cube orientation data and tracking details. The server's role involves receiving HTTP requests from the cube, processing the data, and transforming it into requests sent to the Clockify API.

5.2.3.1 Ktor Framework

To implement this server-side functionality, Ktor is utilized. It is a lightweight asynchronous web framework providing extensibility through plugins. It is built from the ground up using Kotlin, a concise, multiplatform language brought by JetBrains, the creators of Ktor. [28]

5.2.3.2 User Configuration and Project Assignment

The suggested solution also uses the server for user configuration; the user fills out their API key, which is used to access the user's workspaces and projects. The server retrieves the user's projects and sends them to the device, where the user can assign specific projects to each side of the cube. Once the configuration is complete, the cube saves the user configuration to the database or the device memory and waits to be turned to one of its sides.

5.2.3.3 Handling Cube Requests

When the cube is turned to a side with an assigned project, it sends a request to the server. The server then initiates another request to the Clockify API, passing the tracking data along with user details retrieved from the database. The server handles the API response and relays any errors back to the device for display.

Chapter 6

Implementation of the Hardware

This chapter details the practical implementation of the smart cube device, focusing on the integration of hardware components and the wiring of modules necessary for its functionality.

6.1 Wiring Constraints

During the implementation of the smart cube prototype, several key wiring constraints must be considered to ensure proper functionality and protection of electronic components. It is important to keep in mind that the controller unit is based on the ESP8266 microcontroller, not Arduino.

To safeguard the electronic modules and maintain stable voltage levels within the circuit, the implementation incorporates blocking capacitors at the input of each module. They help stabilize the voltage by filtering out noise and fluctuations, protecting sensitive components from potential damage. They also provide decoupling capacitance, minimizing voltage spikes and ensuring consistent power delivery to the connected modules.

Then, pull-up resistors are integrated into the wiring design to maintain a consistent high voltage level for specific signal lines, such as those used for button inputs. They prevent floating inputs, ensuring that the input signal is reliably interpreted as either high or low. They also help reduce electrical noise and interference, enhancing the overall stability of the signal transmission.

In addition to the above-mentioned constraints, it's essential to comply with standard wiring practices to optimize performance and reliability, such as proper grounding and component placement.

The schematic is designed using the KiCAD program, requiring the creation of custom symbols for most of the specialized modules utilized in the project. Standard symbols from the program's library are employed for basic components like capacitors and resistors. The complete schematic diagram detailing the wiring configuration is provided in Attachment A.

6.2 Microcontroller Module

The schematic detailing the MCU setup can be seen in Figure 6.1.

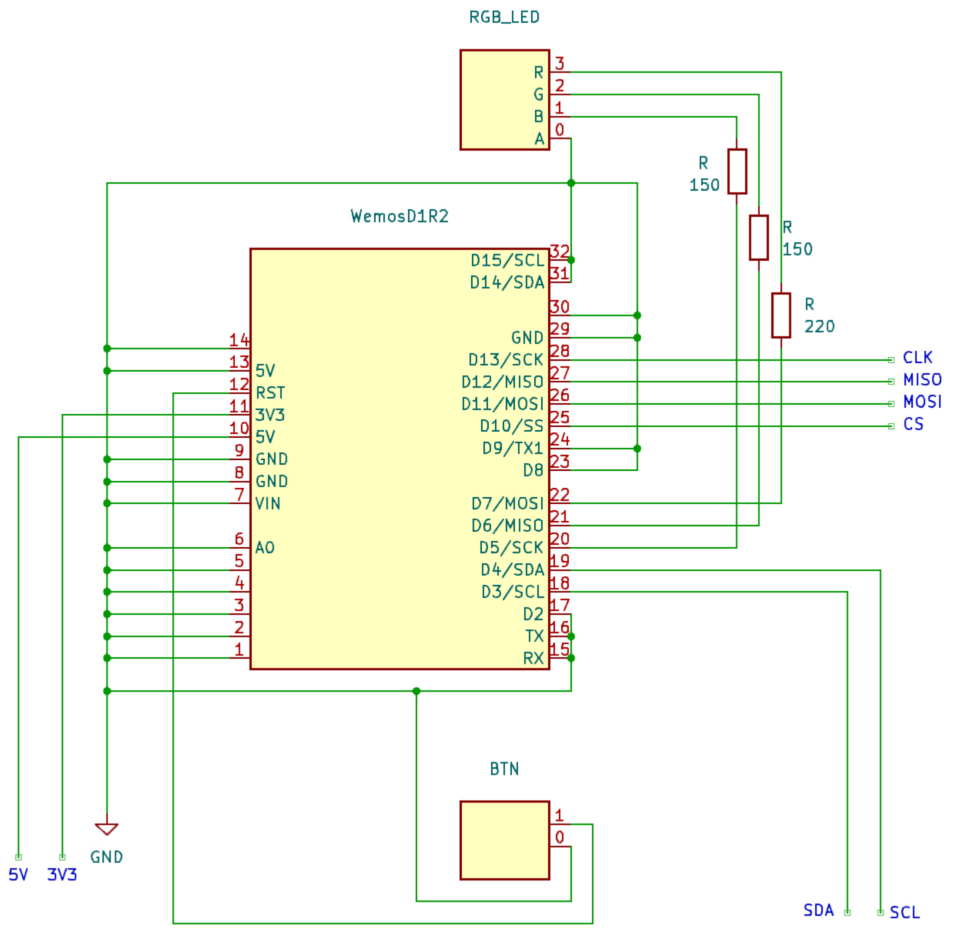


Figure 6.1 MCU Module Wiring Schematic

All of the microcontroller's I/O operates at a standard 3.3V logic level. Notably, the 3V3 pin functions solely as an output, supplying power to connected modules, while the MCU itself is powered via the 5V pin. The I^2C bus is configured programmatically, with the SCL clock signal assigned to pin D3 and the SDA data signal to pin D4. This bus allows the MCU to act as a master device for communication with all connected modules, except for the micro SD card module, which utilizes the SPI bus. The SPI bus is set up with MOSI on pin D11, MISO on pin D12, SCK on pin D13, and the CS line on pin D10.

Furthermore, all ground (GND) and unused pins within the MCU setup are appropriately connected to the ground to ensure stable electrical connections and mitigate potential noise or interference issues. Proper grounding is fundamental to maintaining signal integrity and optimizing overall system performance. The RGB LED is interfaced with pins D5 to D7, while the power/user configuration button is linked to the RST pin for user interaction and device control.

6.3 Gyroscope and Accelerometer Module

The schematic for configuring the GY-521 module can be found in Figure 6.2. This module is powered by a 3.3V supply derived from the MCU, ensuring compliance with the microcontroller's voltage specifications. Communication with the GY-521 module occurs over the I^2C bus, where the module acts as a slave device utilizing the SDA and SCL connections.

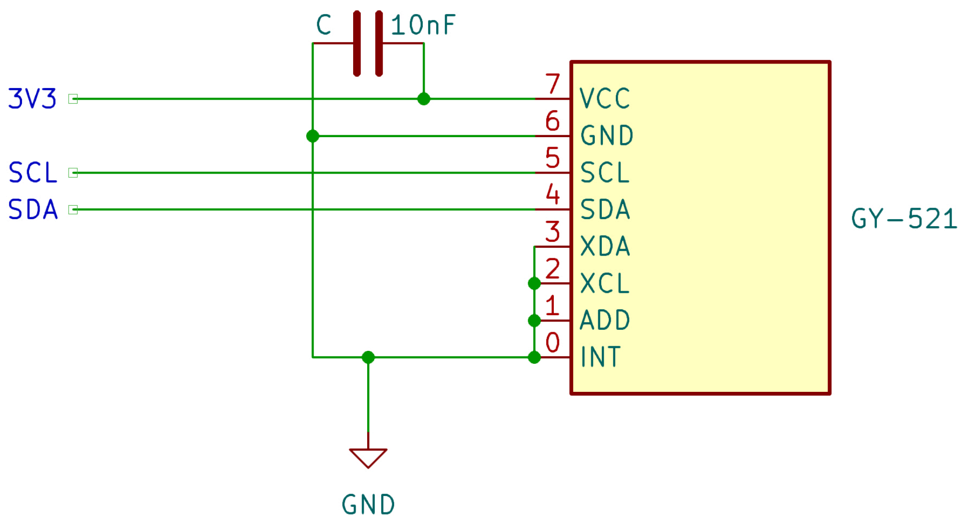


Figure 6.2 Gyroscope and Accelerometer Module Wiring Schematic

To stabilize the power input, a 10nF block capacitor is integrated into the power supply lines of the module. This capacitor aids in filtering and smoothing the voltage supply to minimize noise and ensure reliable operation. Additionally, all GND and unused pins on the GY-521 module are appropriately connected to the ground for effective grounding and signal stability.

6.4 Micro SD Card Module

The schematic of the micro SD card module setup can be seen in Figure 6.3. This module operates at a 3.3V voltage level supplied by the MCU, ensuring compatibility with the microcontroller's specifications. To stabilize the power input and minimize voltage fluctuations, a 10nF block capacitor is strategically placed in the power supply lines of the module.

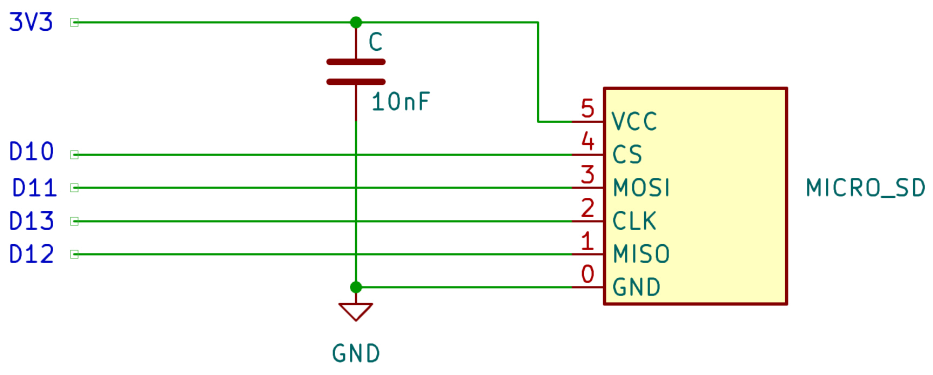


Figure 6.3 Micro SD Card Module Wiring Schematic

Communication between the MCU and the micro SD card module is facilitated through the SPI bus. The module acts as a slave device and utilizes the MISO, MOSI, CS and CLK connections for data exchange.

6.5 Charging Meter Module

The schematic for the charging, including the MAX17048 module is detailed in Figure 6.4. This module operates at a 3.3V voltage level supplied directly from the battery. Communication with the MAX17048 module is facilitated through the I^2C bus, with the module configured as a slave device and utilizing the SDA and SCL wires for data transmission.

The MAX17048 module includes two JST-PH-2 connectors and one of them is used to establish the battery connection with the module. Instead of using one of the JST-SH-4 connectors, direct pins on the module board are employed due to the problem with obtaining the cable with the JST-PH-4 connections. The SDA and SCL pins are connected to the I^2C bus and the ALERT pin is properly grounded since it is not used.

To ensure stable operation and minimize power supply noise, a 10nF block capacitor is strategically integrated into the power supply lines of the MAX17048 module and a separate ground from the rest of the device.

6.6 Charging Module

The schematic for the charging setup, containing the TP4056 module connection, is detailed in Figure 6.4. The B+ and B- pins of the TP4056 module are directly linked to the battery pins. It's crucial not to connect the B- pin to the ground signal used in the rest of the setup to ensure proper operation of the charger's safety circuitry.

The OUT+ and OUT- pins of the TP4056 module are interconnected with a step-up voltage regulator, stabilizing the voltage output to 5V before supplying power to the MCU. A 100nF blocking capacitor is integrated into the circuit to manage voltage stability and filter out potential noise.

Additionally, the IN+ and IN- pins of the TP4056 module interface with the wireless charging module, accompanied by a 10Ω resistor. This resistor is strategically placed to optimize the charging process.

6.7 Wireless Charging Module

The schematic for the wireless charging module setup is included in the Figure 6.4. The outputs are connected to the IN+ and IN- pins of the TP4056 charging module. The output uses a 10Ω resistor to optimize the charging process.

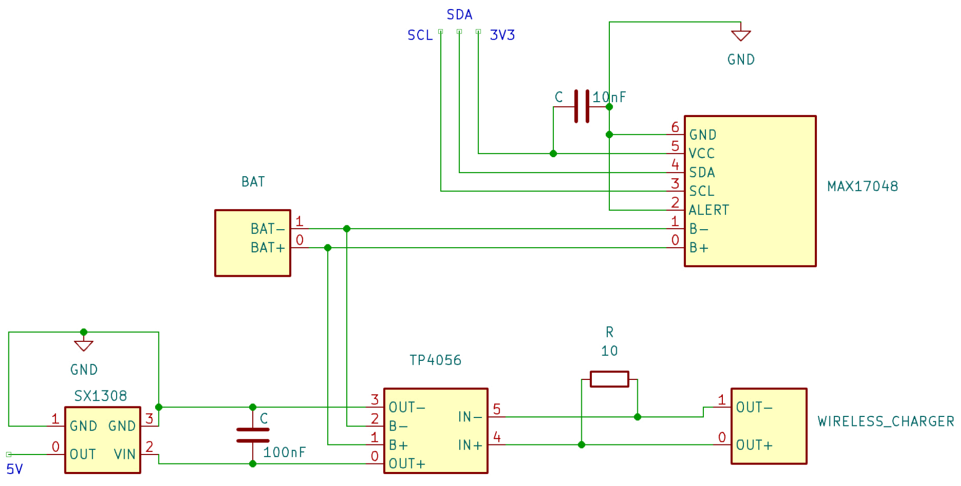


Figure 6.4 Charging Schematic

6.8 PCF8563 Real-Time Clock

The PCF8563 module operates at a 3.3V voltage level and communicates over the I^2C bus. It functions as a slave device using the SDA and SCL wires for data transmission. Additionally, the module features a separate battery backup to maintain timekeeping even when the main MCU is powered down.

The board includes a pin jumper that allows to select the power supply for the PCF8563 module. By setting the jumper to the BAT position, the module is powered by its own dedicated battery, ensuring continuous operation and accurate timekeeping even when the main MCU is inactive.

6.9 RGB LED

A single RGB LED is interfaced with the MCU, with the R signal pin protected by a 220Ω resistor and the G and B pins by a 150Ω resistor. This resistor value is chosen to limit the current flowing through the LED and protect both the LED and the microcontroller from excessive current draw.

The RGB LED typically consists of four pins: one common cathode (or anode) pin and three individual pins for red (R), green (G), and blue (B) colours. Each of these colour pins is connected through a resistor to the corresponding output pins of the MCU (D5, D6, D7 and GND).

6.10 Button

A simple momentary push-button is connected to the RST pin of the MCU. This button serves as a power button and when clicked on a device that has not been set up yet, it triggers the user configuration.

The button is connected in a pull-down configuration, meaning one side of the button is connected to the RST pin to wake the device up from deep sleep mode, and the other side is connected to the ground (GND) through a pull-down resistor (typically 10kΩ). When the button is pressed, it connects the RST pin to the ground, registering a logic LOW state.

6.11 Additional Extensions

There are several potential extensions and enhancements that could be considered to further improve the user experience and functionality of the device.

6.11.1 Wemos D1 Mini Pro

One possible enhancement involves replacing the Wemos D1 R2 development kit with the Wemos D1 Mini Pro. This substitution would not only save space but also enable the creation of a more compact cube design. However, due to the limited number of available I/O pins on the Mini board, the implementation would require integrating an I/O expander into the design to accommodate additional functionalities. Thanks to that the space saved wouldn't be as big, so it was deemed unnecessary for the prototype.

6.11.2 Extra Buttons

Furthermore, the addition of extra buttons presents an opportunity to expand the features and capabilities of the smart cube. For instance, integrating a dedicated button could facilitate Pomodoro timing functionality, allowing users to track and manage work intervals effectively. Another button could be utilized for pausing and resuming time-tracking activities, enhancing flexibility and user control. There could also be a low power mode button to switch the device to low power mode. The MCU would be in deep sleep mode most of the time and would only wake up when it has been turned to a new side. One of the reasons why there is only one button in the prototype design is that the buttons need to be specifically placed on the cube for the user to be able to press them when they are visible and reachable to them.

6.11.3 Qi Standard Wireless Charging

Finally, integrating a wireless charging module that supports the Qi standard would provide added convenience by enabling the cube to be charged on the same Qi-compatible charger used for the user's phone or other devices. This enhancement would significantly benefit the overall user experience. Although this type of module was not used in the prototype, given that the existing charging module was already the most expensive component and the Qi module one would cost even more, incorporating a Qi module could be considered in future iterations once other costs are optimized.

6.12 Prototype Price

The total cost of the prototype amounts to CZK 1113. The breakdown of component prices is detailed in Table 6.1. While it's possible to procure components at lower prices from international e-commerce platforms, all components for this prototype are purchased from local Czech stores. The price listed includes various miscellaneous components categorized as ”Other Components,” encompassing smaller and more affordable items such as step-up regulators, capacitors, resistors, LEDs, and buttons. Please note that the delivery charges for these components are not factored into the total cost.

Table 6.1 Prices of Used Components

Component	**Price in CZK**
Wemos D1 R2	179
GY-521	68
TP4056	19
Wireless Charging Module	339
LiPol Battery 104050	188
MAX17048	98
Micro SD Card Module	22
PCF8563 RTC Module	100
Other Components	100
Total price	**1113**

Chapter 7

Implementation of the Program

This chapter describes the implementation of the device's control program, separated into sections describing the classes dealing with various phases of the device and the libraries used.

7.1 Control Program

The control program for the smart cube is implemented in the *main.cpp* file, which contains the setup and loop functions necessary for initializing and continuously operating the device. Serial communication is also configured in this file for testing purposes.

The *MAX17048* library is utilized to manage the battery charge gauge. This library provides a function to retrieve the current, accurate battery percentage. When the battery charge state drops below 5%, the cube blinks three times in periodical intervals to alert the user. When the battery charge state falls below 1%, the cube turns red, saves the currently tracked data to either the server or the micro SD card, and then enters deep sleep mode.

The *Wire* library is used to configure the I^{2}C bus. Default communication pins are used, but the setup is required for interfacing with the *MAX17048*.

The MCU's deep sleep mode is triggered from the loop function (when the cube is turned to the non-tracking side) or from the `CConfig` class (when the device data is deleted by the user) with a parameter of 0 to initiate an indefinite sleep period.

7.2 User Configuration

This section covers functionalities related to setting up and managing user-specific settings for the smart cube. It involves handling data storage and communication with the user interface through a dedicated `CConfig` class, which implements the following methods:

start initializes user configuration in the device's Flash memory

loadUserConfig loads data from the user configuration

saveUserConfig saves data to the user configuration

onlineSetup handles the initial user setup of online features for the device and the subsequent user configuration

deleteData deletes saved user configuration

Using the *ESPAsyncWebServer* and *DNSServer* libraries, the `CConfig` class generates and presents a configuration form to users via a WiFi access point created by the ESP8266. Upon the initial connection, the user provides essential details such as WiFi SSID, password, and Clockify API key to establish connectivity. They can also assign Clockify projects to specific cube sides. Subsequent connections serve for configuration adjustments, such as providing new WiFi information or disconnecting the cube from their account and resetting its data.

For managing the web server shown to the user these functions from the library are used:

on shows the website

addHandler adds a handler to the website to capture requests

To manage WiFi connections, the *ESP8266WiFi* library provides the following functions:

begin to connect the device to the WiFi network with the configured name and password

setTimeout sets a timeout in case of network connection failures to prevent blocking the main thread

resetSettings deletes all previously configured WiFi network information

To preserve user configuration data across device restarts, the *Preferences* library is utilized. This library offers ESP8266-compatible API functions to store and retrieve key-value pairs in the MCU's Flash memory, ensuring persistent storage of essential settings such as WiFi credentials, Clockify API key, and assigned project IDs. It also holds the value if the cube has already been configured. From the library, the functions used are:

getString retrieves a String value stored against a given key

putString saves a String value against a given key

7.3 Orientation Detection

This section focuses on the specific functionalities related to orientation detection using the gyroscope and accelerometer modules. The `COrientation` class facilitates orientation detection by utilizing data from the gyroscope and accelerometer module. The class is responsible for calculating the pitch and roll of the cube based on sensor readings, ultimately determining the upward-facing side of the cube.

For the readings from the sensor, a *Adafruit_MPU6050* library was used. From the library, the getEvent function is used to obtain values of accelerometer, gyroscope and temperature of type sensor_event_t. Those values are further used to compute the pitch and roll of the cube, to calculate the upward-facing side.

7.3.1 Computation Method

To calculate the orientation of a device using the accelerometer and gyroscope, a common approach involves using a mathematical filter to merge the signals from these sensors. While the Kalman filter offers the most accurate results by minimizing calculation errors, it can be complex to understand and implement. An alternative, simpler method is the complementary filter, which effectively manages both high-pass and low-pass filters simultaneously. This means it filters out high-frequency signals (like accelerometer vibration) and low-frequency signals (such as gyroscope drift). [29]

However, for the smart cube application, neither of these methods has proven to be necessary. The accelerometer alone provides reliable data, with the main issue arising from vibrations that can mimic device movement. [29] Testing conducted by the author showed that significant shaking, equivalent to a major earthquake, would be needed to affect the accelerometer readings noticeably. Moreover, there is a built-in delay before task tracking switches with a cube flip that helps prevent accidental mistracking and cancels out this error.

So the method employed to determine the upward-facing side of the cube relies on straightforward calculations involving rotational forces known also as moment forces, specifically roll and pitch. To define pitch, roll and yaw in linear systems, it is necessary to establish the three primary axes: X, Y, and Z. Since only roll and pitch are utilized for the computations, only the horizontal axes X and Y are focused on. [30]

A roll moment refers to a force that endeavours to rotate a system around its X-axis. Conversely, a pitch moment seeks to rotate a system around its Y-axis. To provide visual clarity, the moments are illustrated in the Figure 7.1. [30]

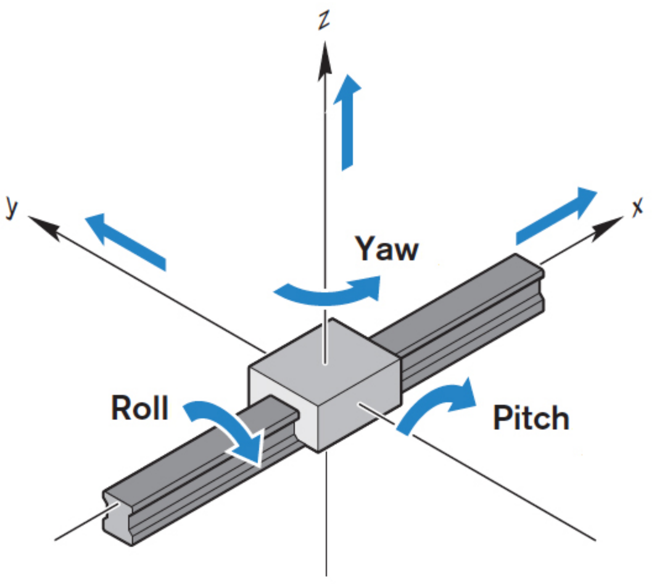


Figure 7.1 Roll, Pitch and Yaw[30]

The MPU6050 module used in the device does not provide the roll or pitch, but it does provide the values for acceleration along each axis, which can be used to calculate both. For the calculation, the gravitational force g is used, since it always acts perpendicular to the Earth's surface. So when an object is tilted at a *theta* angle, part of the force acts along the X-axis, and part acts along the Y-axis, as can be seen in Figure 7.2. [31]

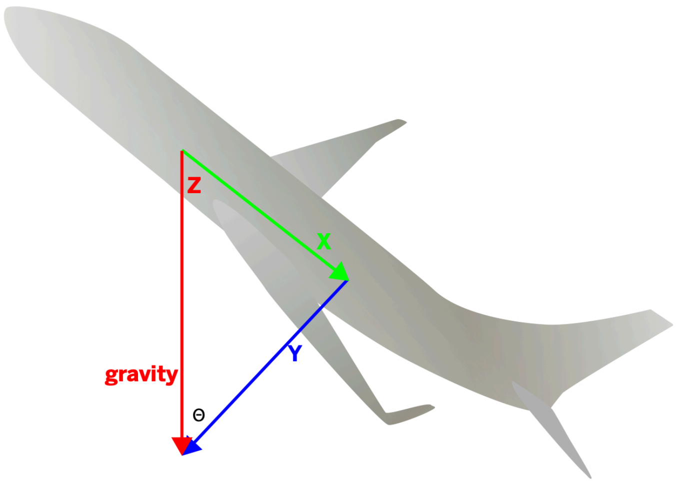


Figure 7.2 Demonstration of the Acceleration Forces [31]

Then the X-axis acceleration a_x and Y-axis acceleration a_y can be determined using trigonometry in the equations in Equation 7.1. [31]

$$a_x = g\sin(\theta), \quad a_y = g\cos(\theta) \tag{7.1}$$

To go from the acceleration values to angles (in radians), more trigonometry is used, with the caution not to divide by zero, using an approximate answer. The results of this transformation can be seen in Equation 7.2 for pitch and Equation 7.3 for roll. [31]

$$pitch = \arctan\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) \tag{7.2}$$

$$roll = \arctan\left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}\right) \tag{7.3}$$

7.3.2 Orientation of the Cube

The following methods are implemented in the class COrientation:

start initializes the COrientation class, setting up necessary configurations for interfacing with the gyroscope and accelerometer modules

getAngle calculates the pitch and roll of the cube using the accelerometer and gyroscope values and saves it to the class variables

chooseUpwardSide determines the upward-facing side of the cube considering the cube's orientation in 3D space and returns it

checkOrientation utilizes the other functions to figure out whether the flip of the cube was accidental or real

checkSwitchDelay checks if the cube has been flipped to the new side for long enough to start tracking a new task

resetSwitchDelay resets the switch delay after a new flip of the cube

chooseColor chooses which colour the cube LED should be switched to

checkEntryType checks if the flip should trigger new entry or finish the running entry and signal the device to turn off

blinkSideColor blinks the colour of the upward-facing side to signal that a new entry could be triggered

7.4 Tracking

The tracking functionalities are encapsulated within the `CTracking` class, which serves as the core component for managing time tracking in both offline and online modes.

In offline mode, `CTracking` handles the local storage of tracked activities when an internet connection is unavailable. It efficiently manages the storage and retrieval of tracked data from the micro SD card, ensuring that all logged activities are securely stored until synchronization with the online platform is possible.

For online mode, `CTracking` interfaces with the server to seamlessly transmit tracked activities to Clockify's API and show it to the user during the usage of the device. This involves establishing a secure connection to the server, enabling real-time synchronization of tracked data with the user's account.

For the work with the PCF8563 RTC module, an *RTClib* class is utilized. From the library, the now function is used to retrieve the current date and time.

Independent of the current tracking mode, these methods are implemented:

start initializes the RTC and micro SD card modules and prepares the file for the tracking data, if the file contains logs from the last session, it triggers the upload to the server

newEntry checks the internet status of the device and decides if the tracking data should be uploaded or saved

7.4.1 Offline Mode

Since the cube is designed to be used in the office environment, it is presumed that a reliable internet connection will be available for the most part, but the cube should be able to operate offline too. This capability is managed through the `CTracking` class, which handles the storage and synchronization of tracking data when offline. The main difference from the online operation is that the tracking data is not sent to Clockify but saved on a micro SD card instead.

To allow that, libraries *SdFat* and *sdios* were used. The originally planned *SD* library is not working properly with the chosen hardware module for an unknown reason, so these were utilized instead. From the *SdFat* library, the function begin is used to initialize the library and card, setting the CS pin (D10 or pin 15 on Wemos D1 R2). A file with the tracking data is created or opened using the open function with the O_WRITE or O_READ, O_CREAT and O_AT_END flags to open the file for write or read, appending the new text at the end and creating the file if it does not exist yet. Then it can be used to write or read the tracking data. Only one file can be opened at the same time, and it needs to be closed after the operation is finished.

Once a connection to WiFi is established, the cube starts automatically syncing unsent data to the server, ensuring data integrity and preventing duplicate records.

saveEntry saves tracking data for a new time entry

endRunningEntry save tracking data for the termination of a running entry

startMicroSD starts the micro SD card communication

createFile creates a new file on the micro SD card

writeFile writes data to the file on the micro SD card

readFile reads data from a file on the micro SD card

7.4.2 Online Mode

The online functionality of the smart cube, managed through the `CTracking` class using HTTP requests, enables real-time tracking and communication with Clockify's server when connected to a WiFi network, which should be possible after the user configuration. This capability ensures immediate updates to project tracking information and seamless interaction with the Clockify platform.

For the WiFi communication, the same functions from the *ESP8266WiFi* library were used as described in section 7.2.

The HTTP requests are managed using the *ESP8266HTTPClient* library. The following functions from the library are used:

begin initiates a new HTTP request

addHeader includes a header in the HTTP request

POST sends the request to the server

end terminates the request

The smart cube initiates HTTP requests to the backend server to switch to the currently tracked side, a process further explained in the subsequent subsubsection 7.4.2.1. During user configuration, all user data is stored in a database linked to a unique cube ID. This approach allows the cube to transmit only its ID, tracked side, and tracking start time to the backend, delegating the rest of the handling to it. Additionally, to prevent accidental switches when the cube is handled, the switch request is delayed to verify the time spent on the new side before execution. Any API request errors returned by the server are communicated back to the cube and indicated to the user through specific light signals, ensuring seamless interaction and user feedback throughout the operation.

The online mode of the device is implemented in the following class methods:

postNewEntry sends a post request with a new running time entry

postOldEntry sends a post request with a finished time entry

7.4.2.1 Server

The server component of the smart cube project is implemented using Kotlin and Ktor, incorporating essential plugins such as Serialization and PostgreSQL to enable robust functionality. This section details the server initialization process, configuration of dependencies, and routing setup to handle time tracking and user configuration functionalities.

The server is initialized as a Kotlin application using Ktor's application factory. The necessary plugins, e.g. Serialization (to handle data serialization) and PostgreSQL (for database connectivity) are added to facilitate efficient data processing and storage. Configuration, including the HTTP port and host, is done to ensure seamless communication with connected devices like the smart cube.

The server was initialized following the instructions within a Kotlin application using Ktor's application factory and the plugins, such as Serialization and PostgreSQL, were added. The configuration was done, including the HTTP port and host setup.

The server's routing system is designed to handle distinct functionalities related to time tracking and user configuration. All control unit requests are of the POST or GET type because the cube only requires the response code to display any potential errors or get the data itself. The server subsequently transforms these requests into the required format.

post("/cubeRotation/cubeId/side/start") initiates a new activity-tracking session without specifying the termination time

post("/stopTimeEntry/end") marks the end of a running activity tracking session, specifying the termination time

post("/cubeRotation/cubeId/side/start/end") adds a completed activity session with start and end times

get("/getAllTasks/cubeId") retrieves detailed task information associated with all of the user's projects

post("/stopTimeEntry/cubeId/end") ends the currently running task, this request is transformed to a PATCH request internally by the server

For the prototype, certain server functions, such as saving user information to the database, were omitted because they were not within the scope of this thesis's objectives. The focus was primarily on implementing core functionalities related to time tracking and integration with Clockify rather than extensive server-side operations involving user data management and storage.

7.5 Cube Behaviour

The behaviour of the cube is implemented in the CCube class. This class contains information about the current state of the cube and its attributes, such as the ID of the cube used for identification on the side of the server and the current and last upward-facing sides of the cube. The following methods are implemented:

setColor sets the color of the RGB LED

loadingBlink blinks the white light from the LED to signal loading

warningBlink blinks the red light three times to signal error

successBlink blinks the green light three times to signal success

Chapter 8

Testing

This chapter provides a comprehensive overview of the testing performed on the prototype and its individual components. The testing process encompassed both individual module testing and integrated system testing using a breadboard setup, as depicted in Figure 8.2. The testing environment utilized a breadboard configuration, facilitating flexible and systematic evaluation of each module's functionality and the overall system performance. This setup allowed for controlled testing of individual components before integrating them into the complete prototype.

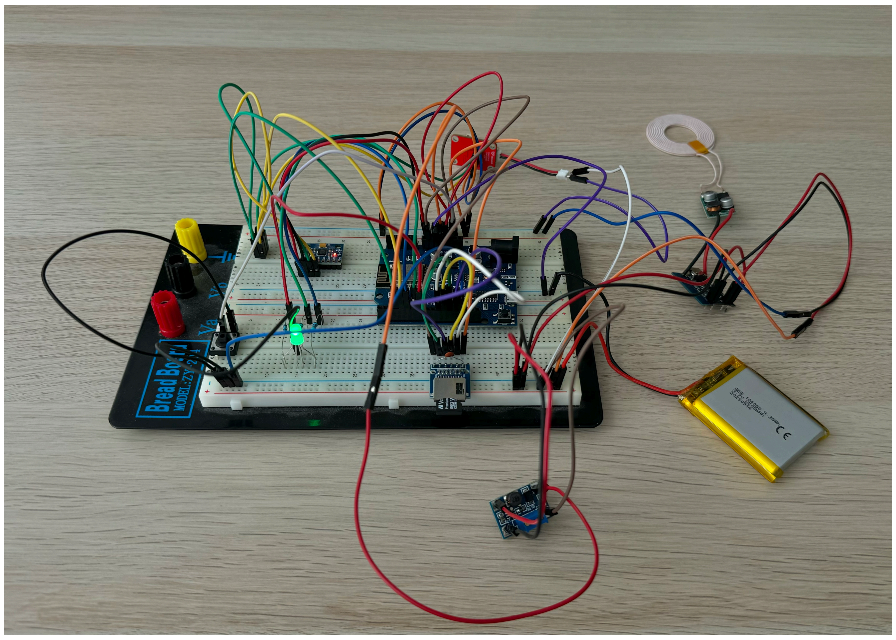


Figure 8.1 Prototype Testing Using a Breadboard

8.1 Module Testing

Each module selected for integration into the prototype underwent adequate testing to evaluate its functionality and compatibility with the MCU. The testing process was performed on a breadboard configuration where the MCU and relevant modules were interconnected and assessed for proper operation.ss

The gyroscope and accelerometer module was initially tested by interfacing with the MCU, which transmitted sensor data and computed results over UART communication to the IDE Serial Monitor tool. Subsequently, a diffusing RGB LED was incorporated into the setup to verify accurate colour switching based on device orientation changes.

The micro SD card module underwent testing by creating a file on the connected micro SD card and writing data from the control program to it. UART communication was employed for this testing phase. Additionally, the micro SD card required formatting in FAT32 format before use to ensure compatibility.

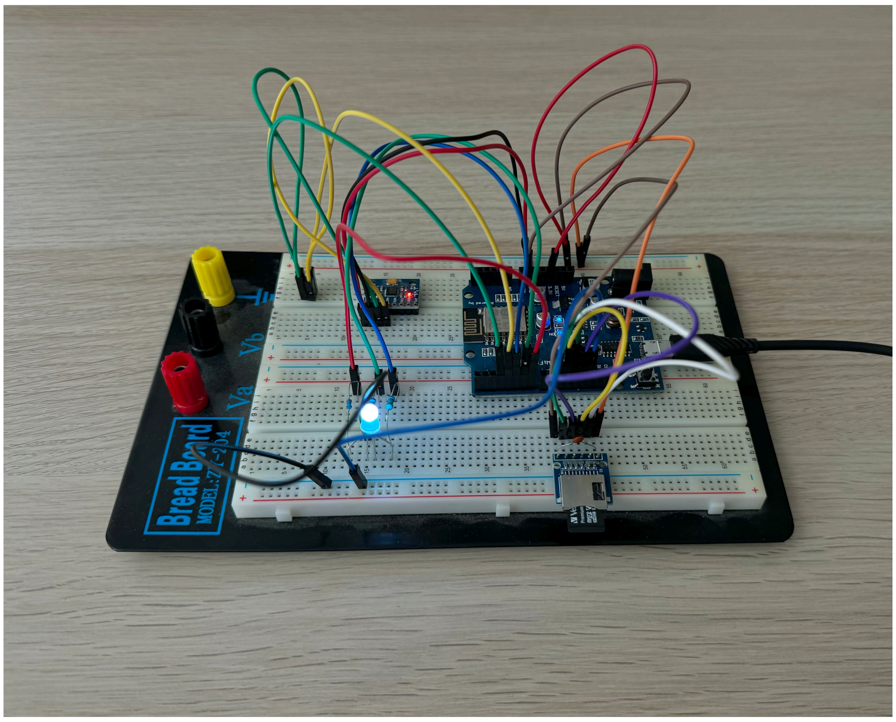
Separately, the functionality of waking from deep sleep using an external button was validated through dedicated testing to confirm its responsiveness and reliability.

The charging system, including the charging circuit, charging meter, and battery, was tested comprehensively as a whole unit. Initially, the batteries were charged using a safety charger connected via a USB Type-C cable, followed by testing with a wireless charging module. Throughout the testing, the charging meter module transmitted battery data over UART communication for monitoring and analysis.

Finally, the fully integrated setup was tested as a whole to ensure seamless interaction and comprehensive functionality across all integrated components. This final testing phase validated the successful operation and interaction of all modules within the prototype, ensuring readiness for subsequent development stages and real-world applications.

8.1.1 Device Orientation Testing

The accelerometer and gyroscope module proved to be sensitive to both slow and fast movements. The switching threshold for each side is set to approximately 45 degrees, making it easy for the tracking to switch as expected by the user. Vigorous shaking of the prototype did result in LED indications of inaccurate tracking changes, but those were successfully mitigated by implementing a delay before finalizing the tracked task switch.



■ **Figure 8.2** Module Testing

8.1.2 Wireless Charging Testing

The wireless charging feature of the prototype was tested to verify its efficiency and reliability. While the charging speed was not very rapid, the testing confirmed that it provides sufficient power for the planned operational cycle when the cube is charged overnight, outside of working hours.

8.2 Program Testing

The implemented program was tested using the Serial Monitor feature within the Arduino IDE over the UART serial bus, with the baud rate set to 115200. The tested features printed test sequences to the IDE.

During the testing, a Ktor server running on a local WiFi network was utilized to communicate with the API. The device communicated as expected via the WiFi network.

8.2.1 Used IDEs

Initially, the development of the time-tracking device prototype began with the Arduino IDE due to its simplicity and familiarity. It provided a straightforward platform for writing and uploading code to the ESP8266 microcontroller used in the prototype.

However, as the complexity of the project increased and more functionalities were added, managing the project within the Arduino IDE became challenging. The project required efficient library management, dependency tracking, and integrated debugging tools, which were not fully supported by the Arduino IDE.

To address these limitations and streamline the development process, the author transitioned from the Arduino IDE to PlatformIO. This transition enabled the author to manage project dependencies efficiently, integrate additional functionalities using external libraries, and leverage advanced debugging tools for troubleshooting and optimizing the code. This shift significantly improved the development workflow and allowed for more flexibility and scalability in the implementation of the time-tracking device prototype.

Chapter 9

Conclusion

The primary goal of this thesis was to develop, implement, and validate a functional prototype of a wireless smart cube designed to simplify time tracking using Clockify's API. The device was constructed on the ESP8266 platform, integrating an MPU6050 gyroscope and accelerometer module to detect tilt and trigger time-tracking actions in Clockify.

Through successful implementation, the device effectively starts, switches and stops task time tracking based on its orientation, providing visual cues for the active project. Furthermore, the cube's setup and configuration utilize WiFi connectivity, while wireless charging capabilities ensure convenient and cable-free operation.

This prototype can be further refined and adapted to suit specific office environments, offering a foundation for future enhancements and student projects. Potential improvements could include enhanced user interfaces, expanded functionality with additional sensors, or integration with other productivity tools thanks to the versatility of the backend.

Appendix A

Schematic

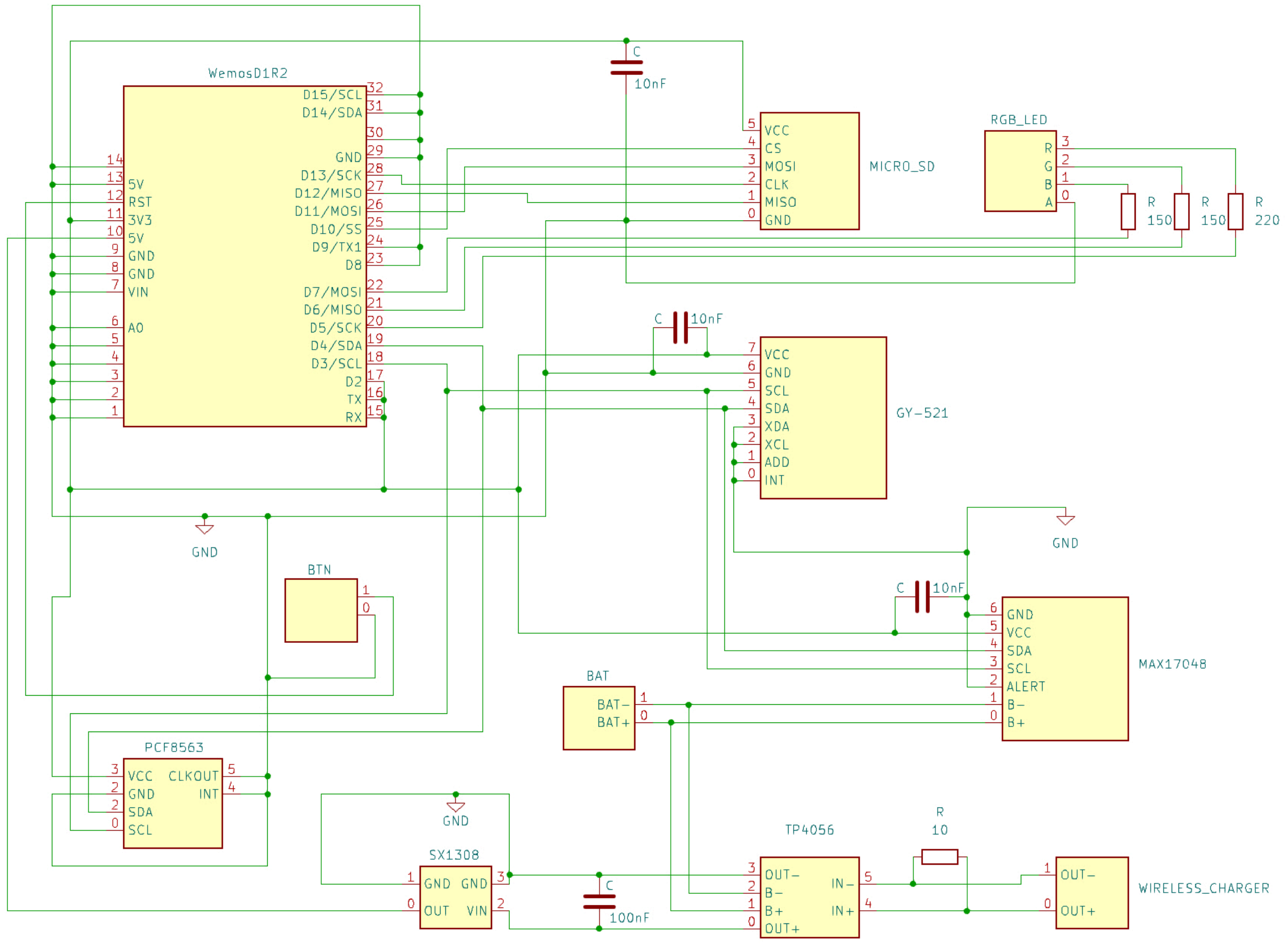


Figure A.1 Wiring Schematic for the Whole Device

Bibliography

1. *TIMEFLIP2: time tracking you'll love! — timeflip.io* [online] [Accessible from `https://timeflip.io`]. [Accessed 11-02-2024].
2. *Time Tracking Cube: The 8-Sided Time Tracker Dice — Timeular — timeular.com* [online] [Accessible from `https://timeular.com/tracker/`]. [Accessed 11-02-2024].
3. *Adafruit IO Time Tracking Cube — learn.adafruit.com* [online] [Accessible from `https://learn.adafruit.com/time-tracking-cube/overview`]. [Accessed 11-02-2024].
4. RAŠIĆ, Ivana Majić. *Why is ESP the most used IoT connectivity MCU — Byte Lab • IoT Development & Production — byte-lab.com* [online] [Accessible from `https://www.byte-lab.com/why-is-esp-the-most-used-iot-connectivity-mcu/`]. [Accessed 07-04-2024].
5. SANTOS, Sara. *ESP32 vs ESP8266 — Pros and Cons - Maker Advisor — makeradvisor.com* [online] [Accessible from `https://makeradvisor.com/esp32-vs-esp8266/`]. [Accessed 07-04-2024].
6. *D1 R2 WiFi ESP8266 — compatible WeMos and Arduino* — botland.store* [online] [Accessible from `https://botland.store/withdrawn-products/6953-d1-r2-wifi-esp8266-compatible-wemos-and-arduino-5904422335298.html`]. [Accessed 03-04-2024].
7. *dratek.cz* [online] [Accessible from `https://dratek.cz/docs/produkty/1/1226/1478466175.pdf`]. [Accessed 03-05-2024].
8. *ESP8266 Pinout Reference: Which GPIO pins should you use? | Random Nerd Tutorials — randomnerdtutorials.com* [online] [Accessible from `https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/`]. [Accessed 03-05-2024].

9. *ESP8266 Deep Sleep with Arduino IDE (NodeMCU) | Random Nerd Tutorials — randomnerdtutorials.com* [online] [Accessible from `https://randomnerdtutorials.com/esp8266-deep-sleep-with-arduino-ide/`]. [Accessed 07-05-2024].

10. *GY-521 With MPU6050 Gyrocope And Accelerometer Module — pajenicko.cz* [online] [Accessible from `https://pajenicko.cz/gyroskop-akcelerometr-gy-521-s-mpu6050-i2c?search=gy-521`]. [Accessed 03-04-2024].

11. HTTPS://WWW.FACEBOOK.COM/VATSBATTERY. *104050 2500mAh 3.7v MSDS battery — VATS BATTERY — vatsbattery.com* [online] [Accessible from `https://www.vatsbattery.com/product/104050-2500mah-3-7v-msds-battery/`]. [Accessed 10-04-2024].

12. R.O., CzechProject spol. s. *Modul pro bezdrátové nabíjení s výstupem 5V 1.5A | dratek.cz | dratek.cz — dratek.cz* [online] [Accessible from `https://dratek.cz/arduino/5010-modul-pro-bezdratove-nabijeni-s-vystupem-5v-1.5a.html`]. [Accessed 03-04-2024].

13. R.O., CzechProject spol. s. *Nabíjecí deska Li-Ion baterií USB-C | dratek.cz | dratek.cz — dratek.cz* [online] [Accessible from `https://dratek.cz/arduino/34679-nabijeci-deska-li-ion-baterii-usb-c.html`]. [Accessed 03-04-2024].

14. *Metody nabíjení baterií, co je CCCV — velofiala.cz* [online] [Accessible from `https://www.velofiala.cz/n/metody-nabijeni-baterii`]. [Accessed 11-04-2024].

15. *GitHub — DoImant/TP4056-Power-Path-PCB: TP4056 Power Sharing PCB. Charge battery despite connected load. — github.com* [online] [Accessible from `https://github.com/DoImant/TP4056-Power-Path-PCB`]. [Accessed 11-04-2024].

16. *GitHub - LaskaKit/MAX17048-Fuel-Gauge — github.com* [online] [Accessible from `https://github.com/LaskaKit/MAX17048-Fuel-Gauge`]. [Accessed 10-04-2024].

17. *Step-Up Voltage Regulator | How it works, Application & Advantages — electricity-magnetism.org* [online] [Accessible from `https://www.electricity-magnetism.org/step-up-voltage-regulator/`]. [Accessed 03-05-2024].

18. LASKAKIT. *Step-up boost měnič s SX1308 2A | LaskaKit — laskakit.cz* [online] [Accessible from `https://www.laskakit.cz/step-up-boost-menic-s-sx1308-2a/`]. [Accessed 03-05-2024].

19. XUKYO. *Using the EEPROM with the ESP8266 • AranaCorp — aranacorp.com* [online] [Accessible from `https://www.aranacorp.com/en/using-the-eeprom-with-the-esp8266/`]. [Accessed 10-04-2024].

20. *Modul čtečka Micro SD karet — SPI modul : H A D E X , spol. s r.o. — hadex.cz* [online] [Accessible from `https://www.hadex.cz/m533-modul-ctecka-micro-sd-karet---spi-modul/`]. [Accessed 24-04-2024].

21. *PCF8563 RTC Board - Waveshare Wiki — waveshare.com* [online] [Accessible from `https://www.waveshare.com/wiki/PCF8563_RTC_Board`]. [Accessed 16-05-2024].

22. CAMPBELL, Scott. *Basics of the SPI Communication Protocol — circuitbasics.com* [online] [Accessible from `https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/`]. [Accessed 15-04-2024].

23. CAMPBELL, Scott. *Basics of UART Communication — circuitbasics.com* [online] [Accessible from `https://www.circuitbasics.com/basics-uart-communication/`]. [Accessed 15-04-2024].

24. CAMPBELL, Scott. *Basics of the I2C Communication Protocol — circuitbasics.com* [online] [Accessible from `https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/`]. [Accessed 15-04-2024].

25. *Software — arduino.cc* [online] [Accessible from `https://www.arduino.cc/en/software`]. [Accessed 18-04-2024].

26. PLATFORMIO. *PlatformIO: Your Gateway to Embedded Software Development Excellence — platformio.org* [online] [Accessible from `https://platformio.org`]. [Accessed 03-05-2024].

27. *GitHub — esp8266/Arduino: ESP8266 core for Arduino — github.com* [online] [Accessible from `https://github.com/esp8266/Arduino`]. [Accessed 19-04-2024].

28. *Ktor: Build Asynchronous Servers and Clients in Kotlin — ktor.io* [online] [Accessible from `https://ktor.io`]. [Accessed 22-04-2024].

29. *Kalman filter vs Complementary filter | Robottini — robottini.altervista.org* [online] [Accessible from `https://robottini.altervista.org/kalman-filter-vs-complementary-filter`]. [Accessed 12-05-2024].

30. COLLINS, Danielle. *Motion basics: How to define roll, pitch, and yaw for linear systems — linearmotiontips.com* [online] [Accessible from `https://www.linearmotiontips.com/motion-basics-how-to-define-roll-pitch-and-yaw-for-linear-systems/`]. [Accessed 11-05-2024].

31. *aatishb.com* [online] [Accessible from `https://aatishb.com/materials/srr/workshop3.pdf`]. [Accessed 11-05-2024].

Contents of the Attached Media

```
readme.txt ........................ description of the medium content
src
  app ............................. implementation source codes
  server .......................... server source codes
  thesis .......................... thesis source code in LATEX format
text .............................. thesis text
  thesis.pdf ...................... thesis text in PDF format
```