Time Tracker

Making Embedded Systems – Final Project

Graeme Gets

2022

# Contents

[Contents 2](#_Toc102728499)

[Figures 2](#_Toc102728500)

[Overview 3](#_Toc102728501)

[Features 3](#_Toc102728502)

[Hardware 3](#_Toc102728503)

[Architectural Diagrams 4](#_Toc102728504)

[Hardware Descriptions 6](#_Toc102728505)

[Accelerometer/ Gyro 6](#_Toc102728506)

[Serial Command Line Interface 6](#_Toc102728507)

[Buttons 6](#_Toc102728508)

[LED 6](#_Toc102728509)

[Battery & Fuel Gauge 6](#_Toc102728510)

[Software Descriptions 7](#_Toc102728511)

[Command Line Interface 7](#_Toc102728512)

[Commands 7](#_Toc102728513)

[Task Time Management 8](#_Toc102728514)

[Task Assignment 8](#_Toc102728515)

[Task Control 8](#_Toc102728516)

[Data logging structure 8](#_Toc102728517)

[Addressable LED’s 8](#_Toc102728518)

[Task State machine. 9](#_Toc102728519)

# Figures

[Figure 1- Hardware Block Diagram 4](#_Toc102640727)

[Figure 2 - Software Block Diagram 4](#_Toc102640728)

[Figure 3 - Software Hierarchy Diagram 5](#_Toc102640729)

# Overview

This is a remake of a product I saw and thought I would like to make it myself. The idea is to use a Dodecahedron to create a physical way to track time spent during the day - In other words, tracking tasks. Each side of the dodecahedron can be assigned a task such as ‘Email’, ‘Coding’, ‘Lunch’, ’meeting’ etc. As you start or stop a task you simply set the Dodecahedron with the task you are about to do facing up.

The system then starts a timer to track the length of the time of the task. This data can then be exported. For the initial project, this will be via serial, but for a complete project this would be via Wifi or Bluetooth to a device or server.

A set of LED’s and sound will be used to attract attention when a tasks allotted time it up.

# Features

* 12 sided Dodecahedron
* Configuration – Via Command line initially
  + Set each face task
  + Set task min/ max time
  + Set End time
* Each side has RGD lighting to indicate:
  + Flash to indicate x number of Mins
  + Task is being timed
  + Task time has ended
  + Task has been paused
* Speaker to play relevant sounds
  + End of time/start/ stop/pause
* Accelerometer & Gyro to determine which side is facing up.
* (Button on each side to Pause/Start)
* (Initially – Radio Comms for Command line - xBee)

# Hardware

|  |  |  |
| --- | --- | --- |
| Name | Description | Notes |
| Processor board | Black Pill STM32F411CEU6 | Chosen for small footprint so that it will fit into the dodecahedron. |
| Accelerometer and Gyro | GY-521 |  |
| Addressable LED’s | WS2812 | Each side will have a 12 LED ring of Addressable LED’s |
| EEPROM |  | For Log storage |
| I2S Amp | Max98357A |  |
| Battery Gauge |  |  |
| RF Serial | Xbee S2C |  |

# Architectural Diagrams

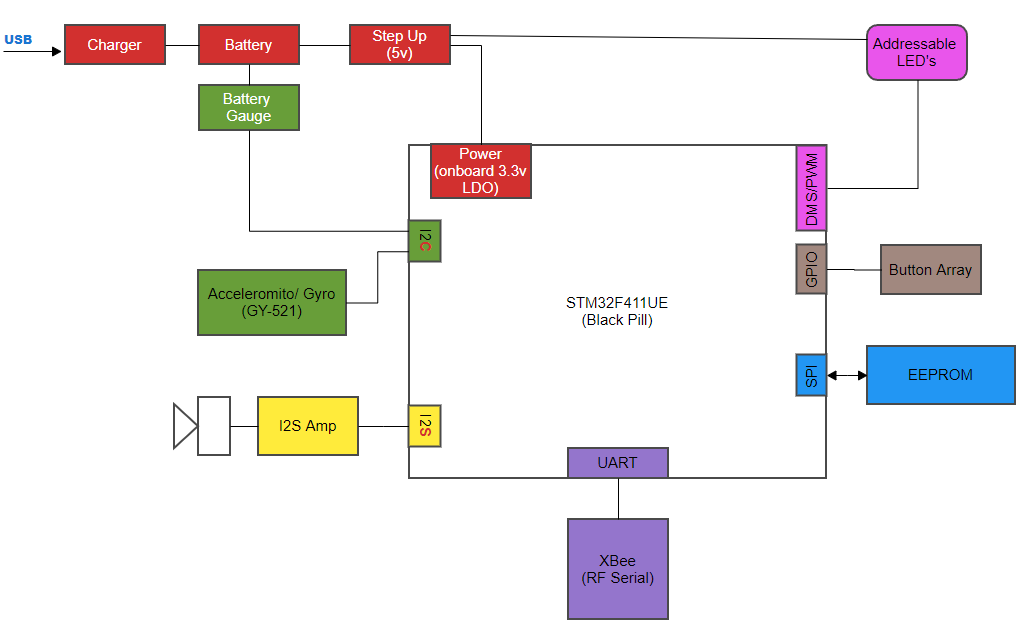


Figure - Hardware Block Diagram

Diagram

Description automatically generated

Figure - Software Block Diagram

Diagram

Description automatically generated

Figure - Software Hierarchy Diagram

# Development Environment and Tools

This project is coded using the STM32Cube IDE and the HAL layer.

## Tools

Multimeter

Saleae logic analyser

# Hardware Descriptions

## Accelerometer/ Gyro

The heart of the project is the detection of the orientation of the dodecahedron linked with the real time logging. The system will need to detect which side of the dodecahedron is facing up and detect a change. The system detects between the specific orientation change and ignores any other movement to cater for the device being moved around the work area or bumped.

The module being used is the GY-521 module which uses the MPU-6050 3 Axis Gyroscope/Accelerometer chip

## Serial Command Line Interface

The device uses a command line interface, implemented over an RF serial radio link. The Xbee will be used as a simple RF serial link. This is to allow for command line management to and from the PC without the need to any cables.

The Xbee radio uses TTL UART for communication protocol.

## Buttons

Each of the 12 sides has a user input in the form of a button or touch button which allows the user to Stop, Start or Pause the timer.

As the only button that matters is the button facing up, the buttons all work in parallel. There is no need for the button on each side to have a specific function.

## I2S Audio Amp

A small Audio amp module using the I2S communication protocol is used for various alerts. These will be small sound bytes in the form of pleasant beeps or tones.

The module is a Max98357A driver from Adafruit (<https://www.adafruit.com/product/3006>)

## LED

The side of the device sports a ring of 12 addressable LED’s. The face up LED’s indicate the following

1. Green – Timer is busy timing the task
2. Blue Flashing – The timer is Paused
3. Red Flashing – The timer is indicating that the allocated time for the task is over

The LED ring is a 12 LED 50mm ring using WS2812B LED’s.

Single red, blue or green colours are used for continuous status in order to reduce the power consumption of the Led’s. If white, for instance, is used at full brightness we can expect there to be a 60 mA draw ***per*** LED. As there are 12 LED rings (one for each side) each with 12 RGB Led’s, the total current draw would be a wopping 144 x 60 mA = 8.64 Amps which is way over the capability of the power system.

However, only the top face is used for the continuous status of the timer limiting the amperage to a maximum of 60 mA x 12 = 720mA. Having just one colour further reduces this to 20 mA x 12 = 240mA.

Keeping the brightness down can further reduce the power consumption.

Side Note : In reality, for a production device of this size its very unlikely that so many LED’s would be used.

## Battery & Fuel Gauge

To Be determined.

The fuel gauge allows the device to monitor the remaining power and let the user know if it needs charging. A slow flashing LED can be used to indicate the need for a charge.

## Power Design & considerations

The system can run off 3.3V for all peripherals except the Addressable LED While the LED rings are designed for 5V operation they appear to work well enough on 3.3v, particularly the logic 1 & 0 levels. However the power supply will need to be able to deliver around 500mA to the LED’s directly to cater for the power consumption. A far lower power rating is achievable if only 2 rinds at most are at full use and brightness.

# Software Descriptions

## Command Line Interface

The system uses a serial command line interface for both system check/debugging as well as task and device management.

Note:

It would be better to use a WiFi or Bluetooth connection with an app to manage the tasks and log the data to a server. However, this project is to prove the embedded software and so the app/web interface can be achieved at a later date.

### Commands

The face ID and the Task ID are the same and are linked when the task is assigned.

|  |  |  |
| --- | --- | --- |
| **Command** | **Parameters** | **Description** |
| List |  | Lists the assigned tasks along with the task parameters and status |
| Assign | Faceid Name MinTaskDuration  MaxTaskDuration  TaskColour | Assigns a task and parameters to a specific face.  Using a faceID of -1 will allocate the task the which ever face is ‘UP’ |
| Stop Task | TaskId |  |
| Start Task | TaskId |  |
| Pause task | TaskId |  |
| Download |  | Downloads the stored task timer data |
| Clear | TaskId | Clears the task timer data.  A Task ID of -1 clears all task data |
|  |  |  |

The base code for the CLI is drawn from Elecia White’s example CLI code from Woko, however the following changed and features have been implemented

1. Use interrupts instead of polling
2. Multi level command structure
3. Actions to deal with control characters in the command string

## Task Time Management

### Task Assignment

Each face of the dodecahedron can be assigned a different task. This ranges from work tasks to entertainment tasks to meal and rest times.

Using the CLI, the user can assign tasks with the following parameters

* Task Name
* Maximum task duration
* Minimum task duration
* Task Colour

A task , which has been assigned, can be started in two ways

1. The dodecahedron is reorientated to with the selected task face pointing up
2. The button is clicked which moves the timer state from Stopped or Paused to started

A user can then either stop or pause the task.

### Task Control

A user is able to pause a task, allowing for a short break.

A Pause can be dealt with in two ways

1. If a face has been assigned a Task of Pause, then turning the device to that face will pause the task.
   1. If the user then moved to a different task the task remains in a paused state until stopped.
   2. If the user moves back to the paused task the task will start again.
2. The top face button is pressed for at least 100ms

A Stop can be dealt with in two ways:

1. If a face has been assigned the Task of Stop, then turning to that face will stop the task.
2. The top face button is pressed for longer than 500ms

**Real Time Clock**

Task timing will be accomplished using the STM32F411’s internal realtime clock.

## Data logging structure

(To Complete)

## Addressable LED’s

Design and Inspiration taken from <https://www.thevfdcollective.com/blog/stm32-and-sk6812-rgbw-led>

The driver used has been designed to cater for a large number of RGB led’s by using only two bytes within the DMA buffer to write duty cycle sts to the timers PWM channel. The driver uses the ‘pulse half complete’ and ‘pulse complete’ interrupts to move new data into the byte that has completed. This allows for a ‘double buffer’ type arrangement which allows for any number of LED’s to be used without the need to create a large PWM buffer to hold the entire ‘byte per bit’ PWM structure.

## Task State machine.

(to be completed)