ASSIST HEIDI

Final Presentation

MARCO POIo

Matthias Finding | ew21b031@technikum-wien.de
David Schmutz | ew21b083@technikum-wien.de
Tjark Neumann | if22x027@technikum-wien.de

https://github.com/Tjark287/marco_poio

Table of contents

01. Problem to solve

02. The Idea

03. System architecture

04. Hardware selection

05. Schematic design

06. Software prerequisites

07. Software architecture

08. Video presentation

09. Design Idea

10. Challanges

11. Lookout

NEVER GET LOST SWIMMING AGAIN

-MARCO POIO

What is the problem to solve?

While blind people oftentimes develop amazing navigational skills on land, when they enter a body of water it becomes very hard to get an idea of where you are due to currents carrying you away in unexpected ways.

NEVER LOSE YOUR CLOTHE AGIAN

-MARCO POIO

How to solve the problem?

The Idea

2-piece device:

- Point of interest (POI)
- Wearable

Navigate with Wearable device to the POI device

System architecture

Overview

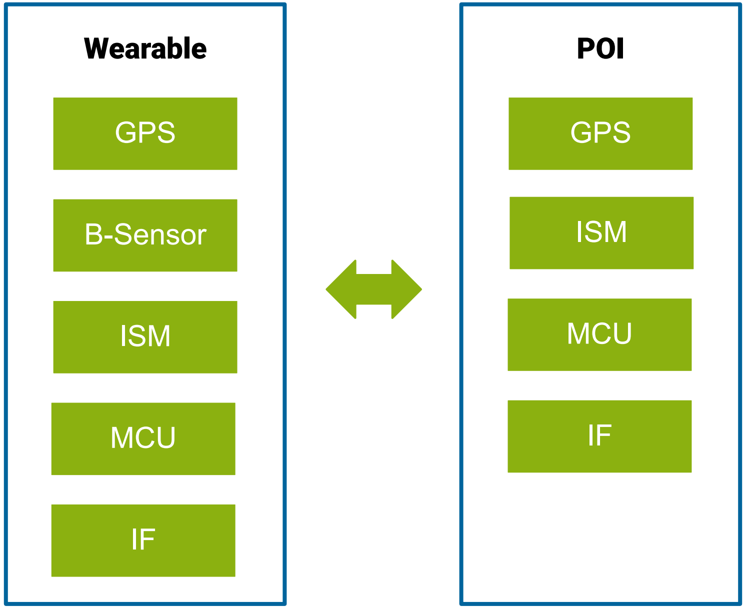
GPS receiver

Compass IC

ISM transceiver

Microcontroller

Interface devices
(Vibration-Motor, Button)



Wearable

GPS

B-Sensor

ISM

MCU

IF

POI

GPS

ISM

MCU

IF

GPS receiver

ISM transceiver

Microcontroller

Interface devices
(Vibration-Motor, Button)

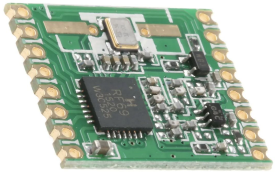
Hardware selection

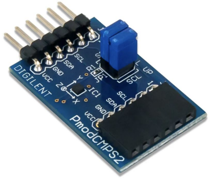
Overview

To implement our system architecture, we chose the following components:

- GPS: Quectel L76-M33 GPS receiver
- ISM: HopeRF RFM69HW 868MHz +20dBm ISM transceiver
- Compass: MMC34160PJ ±16 Gauss magnetic sensor
- MCU: STM32F411CEU 100MHz ARM Cortex M4

All hardware components are 3V compatible



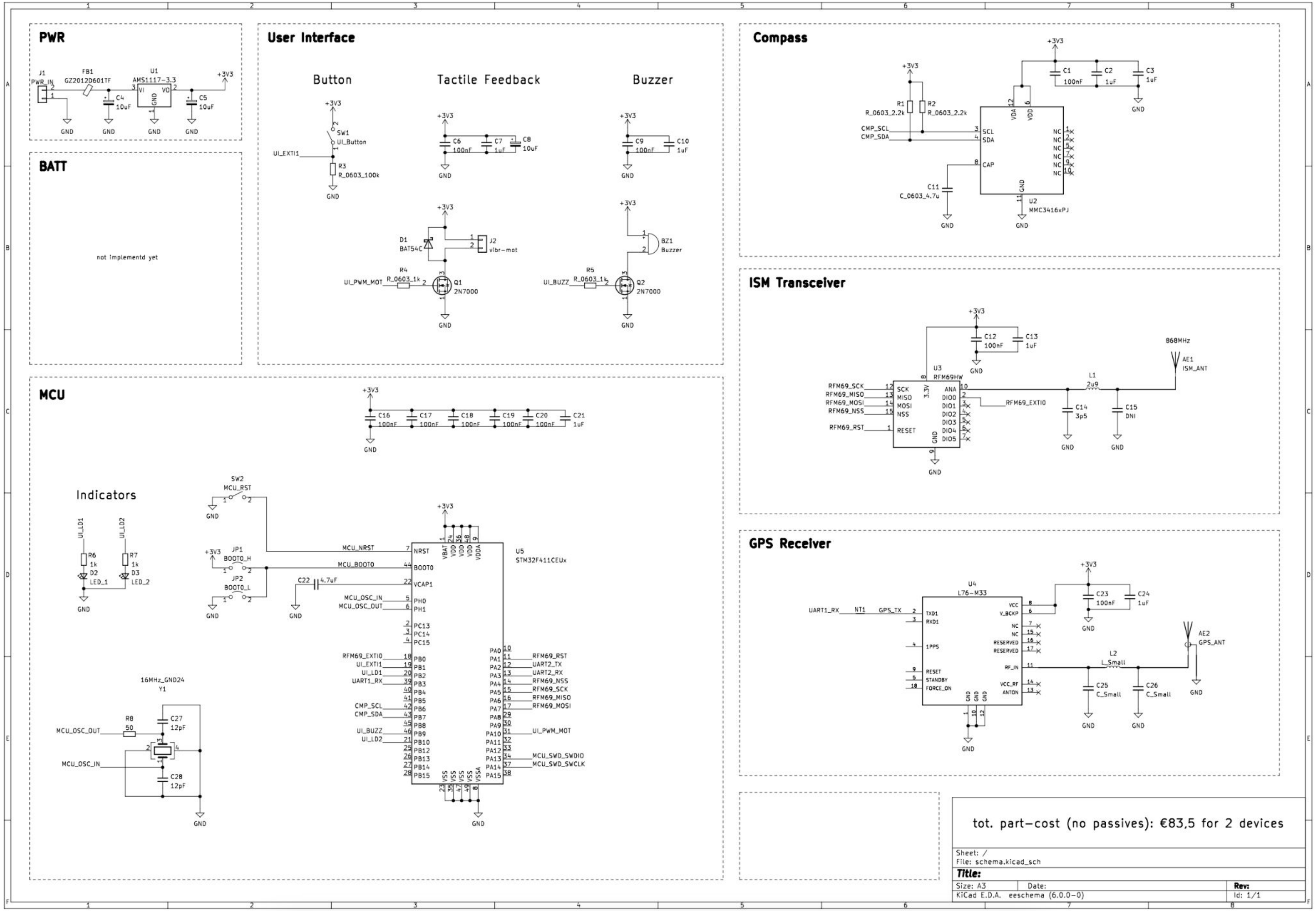


Schematic design

Overview

The schematic describes the connectivity of all hardware components in our design. The main interconnects are:

- GPS: UART interface to STM32 MCU (PB3, receive only)
- ISM: SPI interface + control pins to STM32 MCU (PA1, PA4, PA5, PA6, PA7)
- Compass: I2C interface to STM32 MCU (PB6, PB7)
- User interface:
  - Pushbutton to interrupt pin on STM32 (PB1)
  - PWM control of vibration-motor from STM32 (PA10)
  - GPIO output for Buzzer activation from STM32 (PB9)



tot. part-cost (no passives): €83,5 for 2 devices

Software prerequisites

Overview

External libraries used:

- minmea.c for GPS message decoding
- RFM69.c (ported from C++ to C) to handle the RF communications

Libraries created from scratch:

- helpHeidi.c
- gps.c
- compass.c

Software architecture

Overview

The architecture of the entire system is event- and timer-based. Three main callbacks were implemented:

- **HAL_GPIO_EXTI_Callback**: handles pushbutton input and interrupts from the RFM69 module
- **HAL_UART_RxCpltCallback**: handles information received from the GPS module
- **HAL_TIM_PeriodElapsedCallback**: handles events timed by the main 10Hz system timer

The main loop then carries out all data-send-operations and setting of vibration-motor and buzzer depending on the system state.

To further simplify the coding process, the target device and thereby the system behavior can be defined with a #define statement at the beginning of the code with:

#define IS_POI or **#define** IS_WEARABLE

Video

Prototype

Design

Prerequisites

Accessibility (Cheap)

Braille Integration

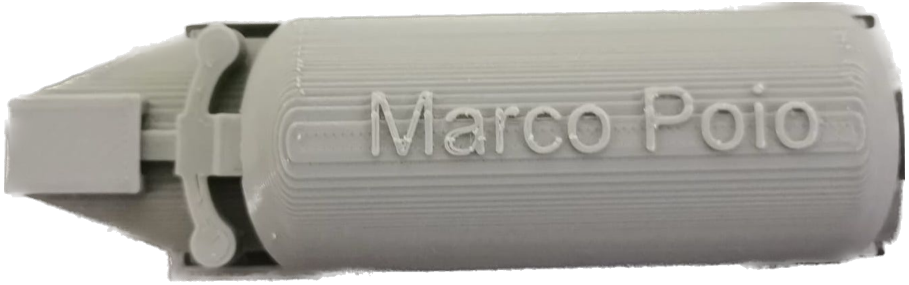
Tactile Feedback

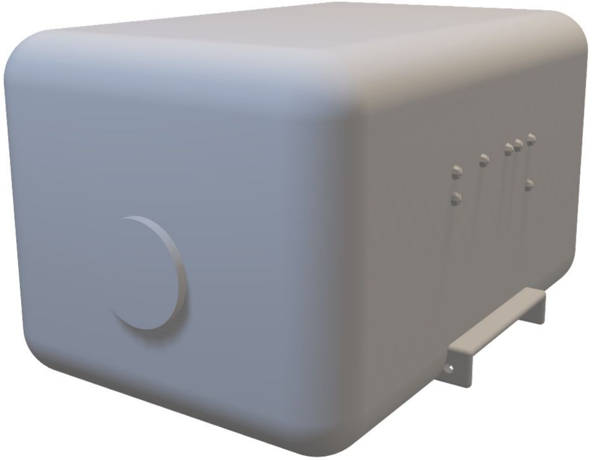
Waterproof

User-Friendly Interface

Design

Idea





Challenges

Software was a monumental task for the time available

SW-Testing was challenging, full system test only possible outside

Budget restrictions led to use of old HW which failed us in the end

Lookout

Next steps

bugfixing

miniaturization of hardware

adding a battery for mobile operation

optimizing power consumption

waterproof casing

wireless charging