Design of Integrated IoT Hardware for Electric Cargo Bike in Tanzania

Juan Angel Luera
MIT EC.793, Fall 2024
Greenfoot Team

Layal Barakat
MIT EC.793, Fall 2024
Greenfoot Team

Olivia McGrath
MIT EC.793, Fall 2024
Greenfoot Team

***Abstract*—This paper documents the work completed by the MIT D-Lab EC.793 student team for Greenfoot Africa's ZELO eTrike. It focuses on the first pass at IoT hardware for production, using an STM32 microcontroller. A brief background of the requirements of the system is given before expanding upon the first prototype of the PCB, including each of the individual components and the STM32 pin setup. This prototype is in the early stages and will require debugging of the board and code, once all hardware has been delivered, to ensure that functional requirements have been met. During the beginning of next year, the design will be integrated with and tested on multiple bikes.**

***Keywords*—printed circuit boards, electric bike, fleet management, IOT system, electronics prototyping**

I. Introduction

Greenfoot Africa was founded in 2019, with the goal of transforming the transport of goods in Africa's urban areas through the increased adoption of clean technology. In 2021, they launched a platform where companies in Arusha, Tanzania and surrounding areas can request delivery or pickup of goods. Over the following 18 months, their team gathered valuable information which led them to develop the ZELO eTrike, a purpose-built electric cargo bike and fleet management system, which is the baseline design for which this work is focused.

The ZELO eTrike, which can carry up to 300 kilos fills a need in the community, where current solutions to transport these loads are to either overload motorbikes and take multiple trips, use a petrol or diesel truck which can carry up to a ton - which is significantly larger than what is needed -, or to leverage public transportation or personal vehicles - the latter of which is restricted to wealthier individuals. It is designed to be manufactured locally, with easy assembly, and it can handle rough roads and diverse climates, travelling over 80 km on a single charge[2].

ZELO eTrike drivers are matched with jobs using an intelligent fleet management and matching system. The eTrike benefits drivers as fuel costs alone of operating a motorcycle or van can be up to 50% of total earnings[2]. Now, drivers are compensated based on distance travelled and payload and do not have to pay for fuel. For the driver-job matching system to function and to determine the appropriate driver compensation, integrated IoT hardware is needed for real-time monitoring and tracking of the bikes.

II. IoT Hardware Development

A previous D-Lab team, working with Greenfoot Africa, designed IoT hardware, leveraging an ESP32 microcontroller. For the updated prototype, an STM32 will replace the ESP32, because of the improved robustness and ease of sourcing parts. The benefit of the ESP32 is the built-in wifi module, but this function is not needed. As the Greenfoot team is looking to expand production, a scalable solution is needed.

A. Goals of IoT Hardware

The purpose of the integrated IoT hardware is to communicate with the fleet management system so that it can assign drivers to jobs. To do this, the requirements of the IoT system are the following:

- Communicate battery and bike health to the fleet management system. This includes current, charge, and temperature.
- Provide real-time location data of each bike.
- Remotely switch the bike on or off for security.
- Record weight of the cargo.
- Log information on the bumpiness of each ride.

The goal is to have a board that can be prepared for the upcoming production scale up.

B. Component Selection

When selecting the parts for the IOT hardware, the current prototype design, which uses an ESP32 microcontroller, the accessibility of products in Tanzania, the project timeline, and the component specifications were considered. The updated prototype includes the following components on a custom PCB.

- NUCLEO L432KC[4] (STM32L432KC) → STM prototyping board
- SIM868 GSM/GPRS/GNSS Hat Module (for Raspberry Pi)[3] → GPS/communications
- MCP-2551[6] → CAN transceiver
- MPU-6050[7] → gyroscope
- SSA-500[5] → current sensor
- TMP36[8] → temperature sensor

The gyroscope (MPU-6050), temperature sensor (TMP36), and CAN transceiver (MCP-2551) are components that the Greenfoot Africa team is currently using for their prototype. The SIM868 was chosen because of its 2G compatibility and easy sourcing for the Greenfoot Africa team. The STM32L432KC is also easily sourced. The use of the NUCLEO prototyping board allows for flexibility during the code generation, debugging, and testing stage of the STM. It will be powered by a 3.3V LDO battery and provide 5V and 3.3V to the PCB. The goal would be to eventually remove the NUCLEO and only add the STM32.

Although we do not need 500A of differential current sensing for the battery at any point (we only expect around 100A at max current draw), the shunt current sensor (SSA-500) is what could be found quickly at D-Lab. The current sensor would primarily be used for the battery and sensing any excess current draw. To be able to monitor battery life and remotely trigger battery shutoffs in case of emergency or theft, the CAN transceiver (MCP-2551) is able to communicate with the battery to do so. The temperature sensor (TMP36) would also be placed near the battery to detect any overheating. All communications would run through the SIM868 Hat Module through 2G networks, such as tracking the bike location and routes or communicating bike battery and health. Roads in and around Arusha are often bumpy, so the gyroscope (MPU-6050) is used to monitor the motion of the cargo box. This, coupled with location data can be used to determine which routes cause more jostling of the cargo and should be avoided when transporting fragile goods.

C. PCB Design

The schematic for our PCB was created in KiCad 8.0. Each component schematic was laid out and connected as per each part's datasheet. Using the NUCLEO board solely allowed us to focus on the connections between the STM and the different sensing components. The components that required I2C pins were connected to I2C pins on the NUCLEO, and components that required UART pins were connected to the RX/TX pins on the NUCLEO.

The SIM868 GSM/GPRS/GNSS Hat Module did not have an explicit footprint, but as it was compatible with the Raspberry Pi extension pinout, that was the footprint used. The current sensor would be connected to the board via a 4-wire connector, so a 4-pin header was used. Every other component had an explicit footprint.

Arranging the PCB was a matter of putting the NUCLEO in the center of the board and placing connected components as close to their NUCLEO pins as possible. Bypass capacitors were placed as close to their respective pins as possible too. 5V and 3.3V power were routed across the board through copper pours, while signal was routed through 0.5mm traces. There was some difficulty in keeping traces short as every component connected to the NUCLEO and there was only so much room on its left and right sides. Future iterations of the board could consider increasing the number of layers for the ground plane or power planes to reduce the number of traces visible.

Many of the passive components (resistors, capacitors) will be directly soldered to the board, as will the MCP-6050 sensor. Notably, the current and temperature sensors will not be. The current sensor itself will be located near the ebike battery, with a 4-pin header soldered to the board to connect to the 4-wire connector. Three lengths of wire will be soldered to the temperature sensor and PCB so it can also be placed near the battery. The NUCLEO and SIM686 board are attached to the PCB via soldered female header pins. This allows for easy removal from the PCB for coding and debugging purposes.

The current limiting factor in terms of cost is the SIM868 board. The SIM868 chip alone is 10x cheaper than the board. The NUCLEO is also double the cost of the STM alone. Future iterations of the board will focus on decoupling sensors from their development boards to reduce costs.

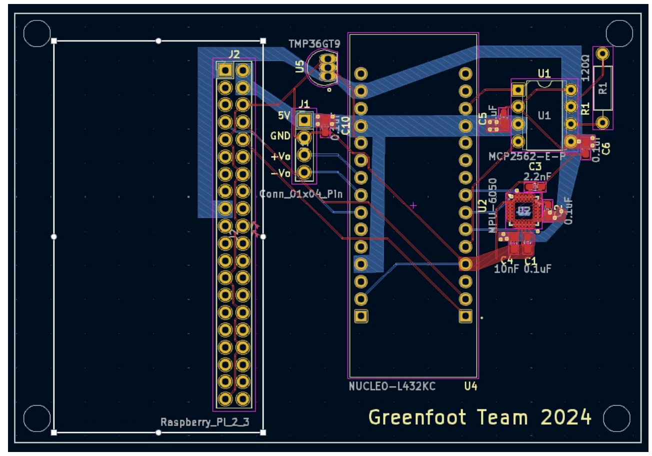


Figure 1. PCB version 1 schematic.

D. Programming the Board

To program the microcontroller, STM32Cube is used. STMCube software is free to download and use, and has the added option of allowing us to assign pins as needed. After loading the STM32L432KC pin layout into the IDE, we used the following diagram of the board to determine which pins have the necessary functions and how to arrange them so that the pins are physically located close to the sensor placement on the PCB.

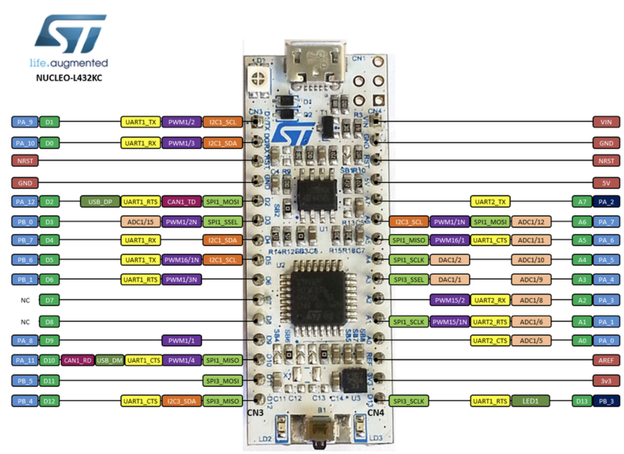


Figure 2. NUCLEO pin layout and available functions.[4]

The current design required that the CAN transceiver use pins PA_11 and PA_12 for CAN RX and TX capability. The differential current sensor required two analog pins, while the temperature sensor required one. The SIM868 TX and RX required UART TX and RX pins and an analog pin. The gyroscope required two I2C SDA and SCL pins. The final STM pin configuration is shown in the following image.

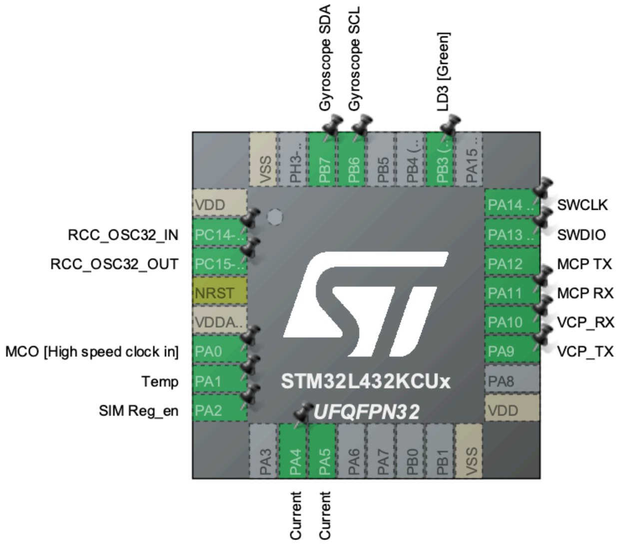


Figure 3. NUCLEO pin setup in STMCube.

There are still pins that remain unassigned on the STM, which include several GPIO pins. These could be used to communicate with an amplifier with input from multiple load cells, if an integrated sensor solution is desired to measure cargo weight.

For the temperature sensor, which arrived early enough to test, the code has required a substantial amount of debugging, so a large effort to get the remaining sensors working is anticipated.

IV. Software Infrastructure

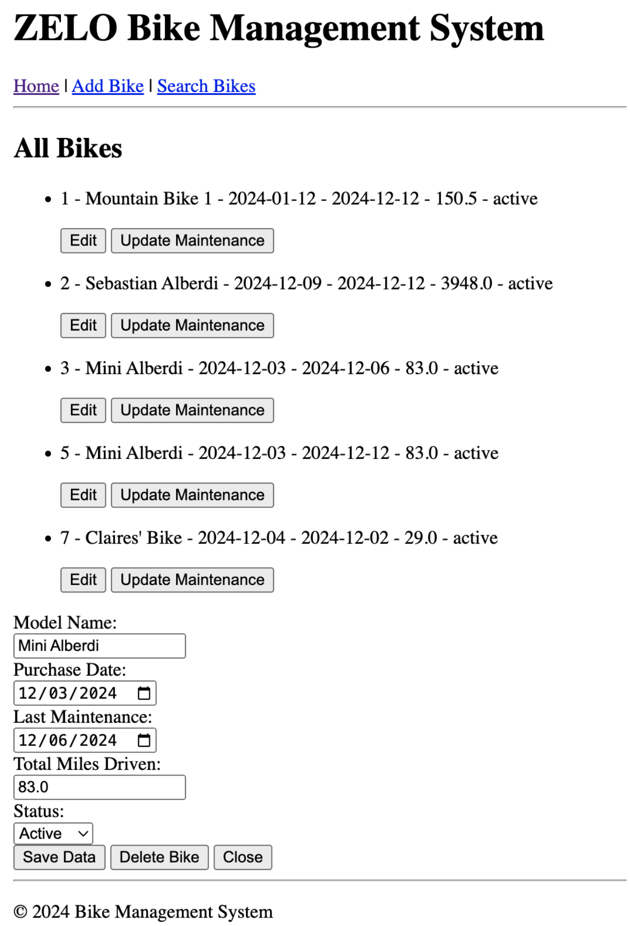
A big challenge that we faced during this project was creating the infrastructure for communication. While the STM32 system is able to transmit data this is useless without proper infrastructure.

A. MQTT

For communication between the microcontrollers and our backend we used the MQTT protocol. First tested this in software by writing code in python for publishing fake gps data in public MQTT channels. Next, we wrote C code that communicated an STM to the MQTT channels through ethernet. (See appendix for more details)

B. Zelo Website

Something else we built was the backend of the website. Since the goal is to eventually be able to use the Zelo bikes akin to an uber you need a centralized data system. We wanted the website to be as simple as possible and quick for deployment, therefore we decided to use Flask, HTMX, SQLAlchemy with Sqlite. These technologies enabled us to create a lightweight app that can easily be deployed. Furthermore, by minimizing the number of languages we reduce the complexity of the site itself. Lastly, using a well known language such as python makes it even more accessible. Our goal with this website is that it can be easily passed down to following teams and even be managed by the people in charge of the start-up.



V. Conclusions

With only one semester to work on this project until now, we have been very time limited. As a result, there is still some work that we plan on accomplishing before travelling to Tanzania and once we are at Greenfoot Africa.

A. Future Work

We have placed the order for our prototype PCB, which will be arriving a couple days after the last class. Once the board and the remainder of our components have arrived, we would like to assemble and test the PCB. Once assembled, we will begin to work on getting readouts from each of the sensors.

The Greenfoot team has stated that they are working to finish two bikes, in addition to their one testing platform, so there should be three ZELO bikes available to work on by the time our team arrives. When packing for the trip we would like to plan to bring at least three sets of hardware so that multiple units can be tested at once. Along with the testing hardware, we will determine where to mount the hardware once the bike and the necessary housing to keep components shielded from the environment.

The current PCB design uses a CAN transceiver to interact with the battery. Only recently was the Greenfoot team able to acquire a datasheet for the BMS, so in future PCB iterations, it might be possible to eliminate the transceiver, as well as the temperature sensor. The BMS datasheet shows that it should be able to return a temperature measurement. This would help to further simplify the PCB design.

Finally, additional options for load sensing will continue to be explored when the team receives a clearer picture of how the cargo box is attached to the bike. Load cells with the capability to measure up to 300 kilos, and any load cell with a maximum load of 100 kilos or greater which can be used in parallel, are costly and could require additional parts to be added to the cargo box. Over the next couple weeks we will explore the potential to create our own load cells, customized for the setup, or the use of thin film strain gages on existing parts of the cargo box.

B. Key Takeaways

There are several takeaways from this project as it stands now. Firsty, it is very helpful to create a development board, especially in the process of testing out different blocks of code and debugging sensors. In addition, there are several ways of laying out a PCB that might work, and the key is to try to move things around until trace lengths are minimized. Finally, time is certainly limited for projects such as these and when it comes to prototyping, things always take longer than anticipated. Lead times for parts are often long, and even getting some bits of information took several weeks to months. Flexibility and adaptability are key.

Acknowledgment

We would like to thank Heewon Lee and Adi Mehrota, as well as Johnson Jacka from the Greenfoot Africa team, for their support this semester.

References

[1] Greenfoot & Access to Energy Institute. "E-Mobility in Tanzania: Business and Technical Insights on Productive Use Cases. Business Models Part 1: Flat Fee Rental for Motorcycle Taxis". 2021.
https://a2ei.org/resources/uploads/2021/10/A2EI_Greenfooot_E-Mobility-Adoption-in-Tanzania_Business-and-technical-insights-on-productive-use-cases.pdf

[2] Greenfoot. "About Greenfoot Africa: Driving Africa's Transition to Sustainable Urban Logistics". 2024.

[3] SIM868 GSM/GPRS/GNSS Hat Module Wiki: https://www.waveshare.com/wiki/GSM/GPRS/GNSS_HAT

[4] STM32L432KC Pins (NUCLEO-L432KC): https://os.mbed.com/platforms/ST-Nucleo-L432KC/

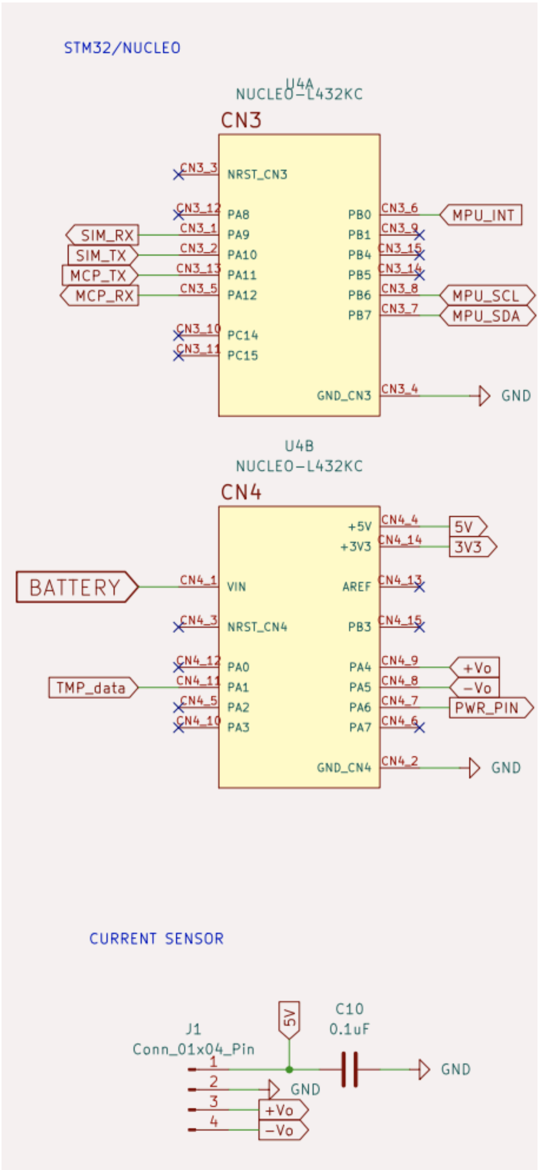
[5] Rideon SSA-500 Current Sensor Datasheet: https://riedon.com/media/pdf/SSA2.pdf

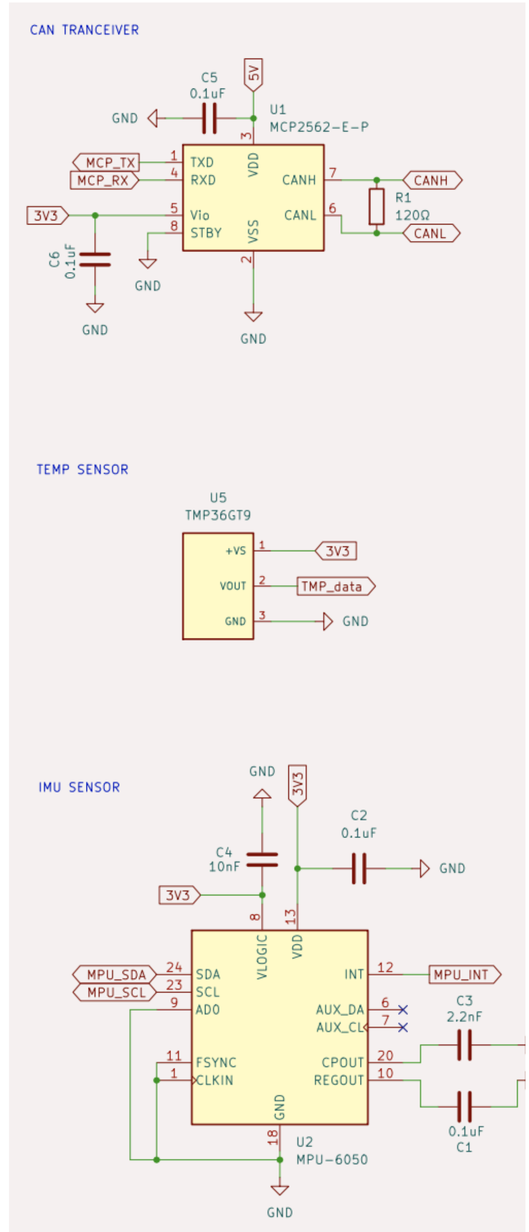
[6] MCP2551 High-Speed CAN Transceiver Datasheet: https://ww1.microchip.com/downloads/en/DeviceDoc/20001667G.pdf

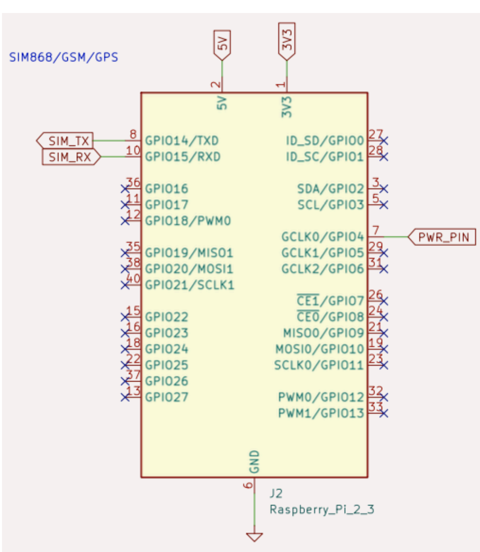
[7] MPU-6050 gyroscope datasheet: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

[8] TMP36 Temperature sensor datasheet: https://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf

Appendix A


STM32/NUCLEO
U4A
NUCLEO-L432KC
CN3
NRST_CN3
PA8
PA9
PA10
PA11
PA12
PC14
PC15
PB0
PB1
PB4
PB5
PB6
PB7
GND_CN3
SIM_RX
SIM_TX
MCP_TX
MCP_RX
MPU_INT
MPU_SCL
MPU_SDA
GND
U4B
NUCLEO-L432KC
CN4
+5V
+3V3
VIN
AREF
NRST_CN4
PB3
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
GND_CN4
BATTERY
TMP_data
5V
3V3
+Vo
-Vo
PWR_PIN
GND
CURRENT SENSOR
J1
Conn_01x04_Pin
5V
C10
0.1uF
GND
+Vo
-Vo


CAN TRANCEIVER
C5
0.1uF
5V
GND
U1
MCP2562-E-P
MCP_TX
MCP_RX
TXD
RXD
VDD
CANH
CANL
Vio
STBY
VSS
3V3
C6
0.1uF
R1
120Ω
GND
TEMP SENSOR
U5
TMP36GT9
+VS
VOUT
GND
3V3
TMP_data
GND
IMU SENSOR
GND
3V3
C2
0.1uF
C4
10nF
3V3
VLOGIC
VDD
MPU_SDA
MPU_SCL
SDA
SCL
AD0
INT
MPU_INT
AUX_DA
AUX_CL
C3
2.2nF
FSYNC
CLKIN
CPOUT
REGOUT
GND
0.1uF
C1
U2
MPU-6050
GND


SIM868/GSM/GPS
5V
3V3
SIM_TX
SIM_RX
GPIO14/TXD
GPIO15/RXD
GPIO16
GPIO17
GPIO18/PWM0
GPIO19/MISO1
GPIO20/MOSI1
GPIO21/SCLK1
GPIO22
GPIO23
GPIO24
GPIO25
GPIO26
GPIO27
ID_SD/GPIO0
ID_SC/GPIO1
SDA/GPIO2
SCL/GPIO3
GCLK0/GPIO4
GCLK1/GPIO5
GCLK2/GPIO6
CE1/GPIO7
CE0/GPIO8
MISO0/GPIO9
MOSI0/GPIO10
SCLK0/GPIO11
PWM0/GPIO12
PWM1/GPIO13
PWR_PIN
GND
J2
Raspberry_Pi_2_3
GND

Appendix B (Zelo Website)

Links to github