Huawei University Challenge 2020

introducing

ProHelmet

a smart Head Up Display to improve safety of motorbike riders

Crafted by Luca Ceragioli

Acknowledgements

Since good ideas never come alone

I wish to thank everyone that has provided me even a little help in this adventure.

Navid Valente, med student, gave me a tremendous help in focusing the screen using a lens. Without his deep knowledge of the human eye this entire project wouldn't have come to light.

Daniele Leonardis, researcher at the Perception Lab of Bio Robotics in Pisa, handed me a free accelerometer when I destroyed the previous one with an overvoltage ten days before the deadline. Thanks to that sensor I was able to complete the last part of the work.

Mattia Biavati, motorbike rider but also med student, gave me a lot of useful suggestions about how to improve the look and the functionality of the helmet. Many aspects of the final prototype bear his style.

Tommaso Burlon, engineering student, made some useful python scripts to convert images and fonts to C array bitmaps.

Fabio Tessaro, motorbike rider and mechanical eng. student, lent me his motorbike to tests the helmet and to shoot a small clip.

Davide Bettarini and **Christian Perissutti**, engineering students, helped me to shoot a large part of the videoclip.

Alessia Sabatino, law student, supported me from the very beginning defining the whole project and she also made the courtesy of wearing the helmet in the videoclip.

I also have to thank the whole **Linux community,** which made possible to have such a wonderful operating system as Debian is; I coudn't even imagine me developing software using a different os.

Free Software will lead to a better world.

Contents

Table of Contents

Long-awaited safety improvements

Safety concerning motorbike riders is crucial, since they have the **highest deaths/accidents rate** compared with pedestrians and car drivers. In 2015, at least 3.939 riders (drivers and passengers) of motorbikes were killed in the EU in road accidents.

According to the US General Council of Safety, over 25% of accidents on the road happened because of **distracted driving**. In motorbikes, the speedometer is attached to the handlebar, completely out of sight if the driver is looking at the road. In addition, even the smartphone, which usually act as gps, is fixed to the handlebar, potentially leading to dangerous distractions.

Since, in order to stare at the motorbike's display, riders are forced too look down they lose their focus on the road. Given that I tried to improve safety on the road for motorbike riders through a **smart HUD embodied in the helmet**.

ProHelmet: a feasible solution

This outstanding helmet provides an easy interface to the driver, telling all the necessary informations such as speed and safety distance as well as driving direction. Similar prototypes have an estimated price of thousands of euros, while this one, even if it is only a proof of concept, has a very **low price** of roughly 40€.

The key is a small 0.96 Inch **oled display** focused through a small **lense**, and driven by a powerful STM32 **Cortex M4 processor.**

The helmet constantly exchanges information with the **driver's smartphone**, using bluetooth communication. This approach allows the helmet to have access to gps and notifications. It is even possible to setup driving directions from the smartphone and, subsequently, to send them to helmet.

The **battery** currently being used to power the whole system is a *cellphone power bank*, that can be easily attached using a usb cable. The power bank fits nicely into the driver's pocket, even thogh in the future a **small battery** may be embodied in the helmet, removing the need of an external power source.

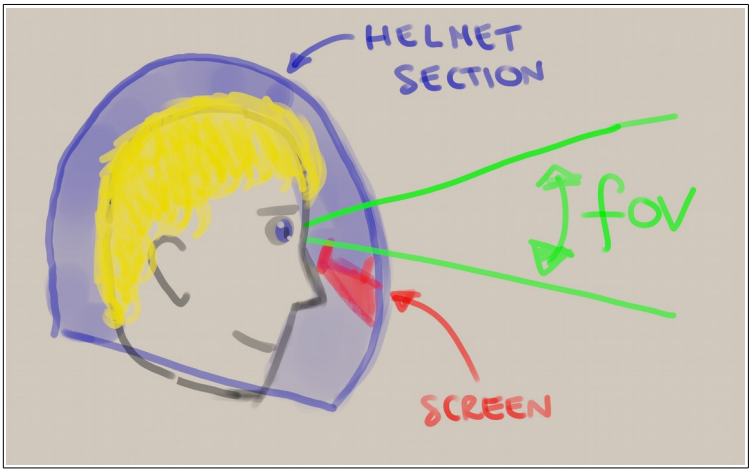
Features:

- Speed-accuracy improved by an embedded accelerometer
- Adaptive real-time stopping distance calculation based on speed and braking force
- Automatic screen brightness adjustment with a photoresistor (backlight control)
- Visual driving directions using text and images
- On-screen push notifications (messages and calls)
- External temperature measurment (ice warning)*
- External barometric pressure check (storm warning)*
- Three buttons in a confortable position
- A custom Android application feeds gps data to the helmet over bluetooth

* Even though temperature and pressure are constantly measured by the helmet, those warnings are not yet implemented because extensive tests are needed. However both pressure and temperature can be displayed on screen if requested by the user.

How to wear the helmet

In order to not block the sight of the road, the screen must be locate sightly below the driver's eye. This solution ensures that the driver has the necessary **field of view** (fov) looking towards.

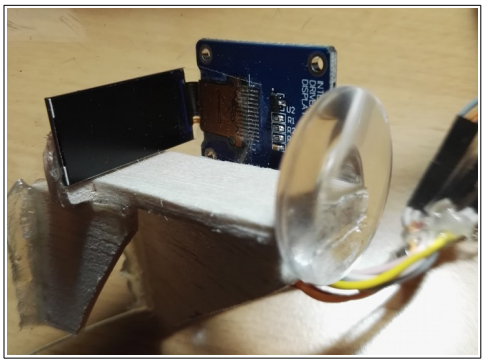


Lens details

The first that problem arises when a screen is placed near the human eye is focusing: in order to focus something without difficulty it must be placed far from the eye, because of the limitations of the human crystalline. In order to create a virtual image far from the eye a lens is used, through the formula:

$$\frac{1}{screen\,distance}+\frac{1}{virtual\,image\,distance}=\frac{1}{focal\,lenght}$$

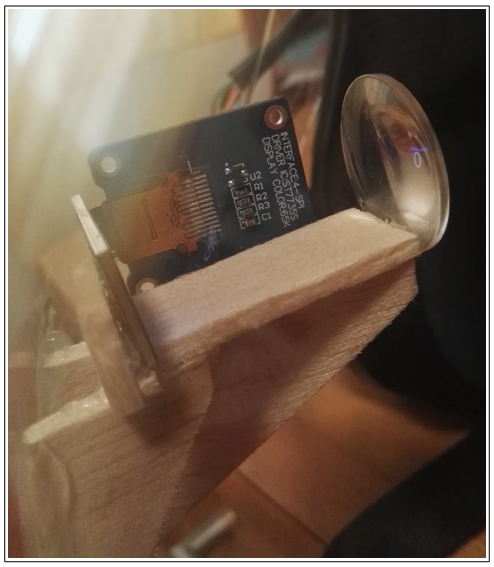
It turns out that the farthest virtual image can be obtained if the screen is placed exactly in the focus. I've chosen a **custom biconvex plastic lens** with a diameter of 30mm and a **focal lenght** of 35mm. Since the space inside the helmet is limited it is important to have a lens with a very short focal lenght, otherwise there will not be enough room inside the helmet to fit the lens and the screen.



The lens is positioned just below the right eye, in this location it is possibile to stare at the screen just by sightly lowering your gaze, without completely losing the sight of the road as would happen if looking down to the handlebar of the motorbike.

As you may see from the picture the lense is held at a fixed distance from the screen using a wooden structure.

Fun fact: it seems that no glue on earth is capable of proving a strong grip over the plastic shield of the helmet. In order to keep the screen steady I used a mix of epoxy and strong liquid putty.

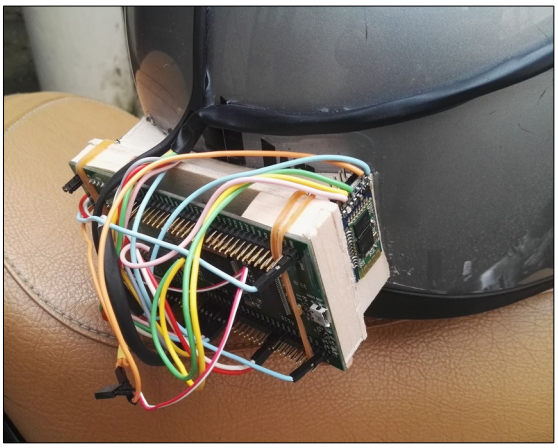


Here is also a picture of the screen-lens structure seen from another perspective.

Question: the screen may be an obstacle in the driver's sight, how do you cope with that?

The lens and the screen should not be in front of the driver's eye but sightly below it. This way, the rider can just lower its glare to look into the screen.

Structural details



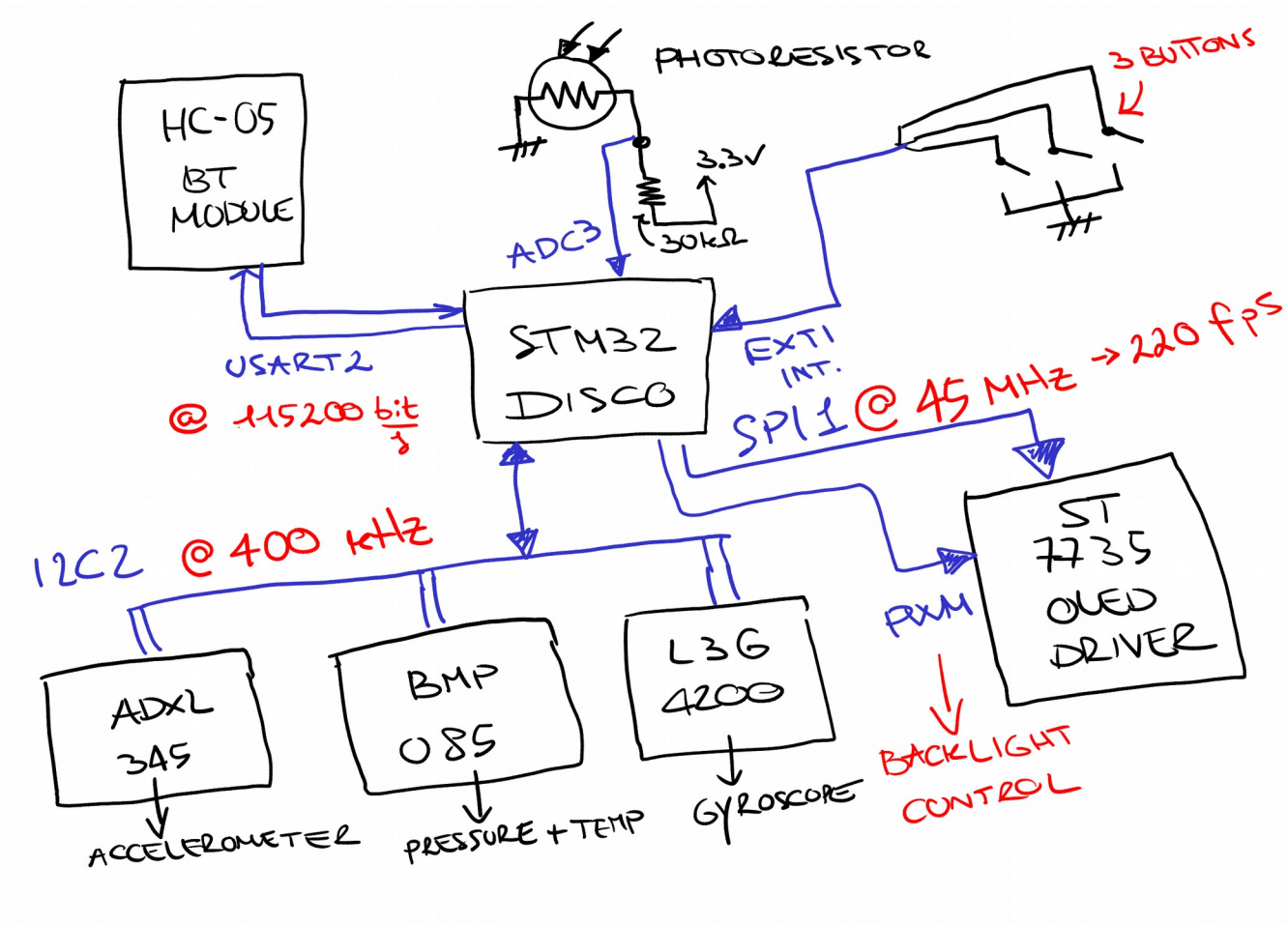
The whole structure is made of balsa, an ultra-light wood that brought an enourmous flexibility during the construction. It is relevant to note that the whole system is not waterproof, although the wiring uses most of the space and thus it would be possibile to integrate all the connection inside the helmet, using thinner wires. The longest wires are tied together using **heat-shrink tube**, otherwise the wires connecting the screen to the discovery board would be messing around.

In addition, in the event of crash, the balsa structure is not hard and it won't harm who is wearing the helmet.

The Discovery Board is attached to the wooden **frame** using two ribbon bands, allowing a quick replacement.

Buttons' location

Brief schematic overview



All the connection are made using thin gauge wire and the connectors are flat pins 1/4".

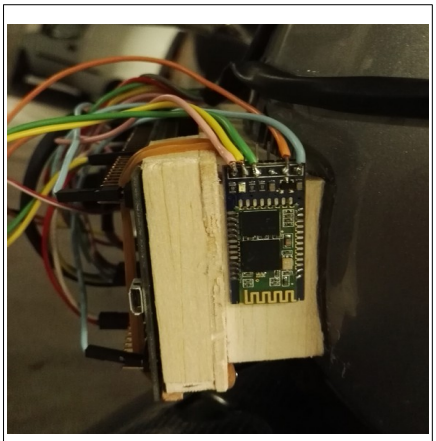
The the oled screen is clocked at 45Mhz using SPI in order to achieve a **smooth FPS** (frame per second) rate and the backlight is controlled using a pwm signal operating at 10 kHz.

Moreover, there are also three buttons located near the helmet boundary, a confortable position, where usually even the lever to lower sun shield lies. All three buttons feature a **software debounce** control.

Photoresistor

The photoresistor, placed over the top of the helmet, is used to adapt the backlight to the effective light condition in order to avoid any hassle or dazzle when entering a tunnel or riding in the dark.

Meanwhile, the bluetooth module is used to communicate with a smartphone to provide notifications and driving directions. This module can be configured using AT commands.



HC-05 Bluetooth module

Software operation

Feel free to check the source code out:

```
$git clone https://github.com/VirtuContraFurore/ProHelmet
```

Source code is roughly 3'500 lines of C files and 1'700 lines of H files (you can `$find . -name '*.c' | xargs wc -l` in the root of the project)

Question: that's a lot of code! Why didn't you used pre-made drivers?

Since I simply cannot trust any driver I haven't checked with my own board and I hadn't much time to put everything together, I toggled each register directly. I leave silly things like "HAL" to little kids using a microcontroller for their first time [sarcasm was used here, via Real Programmers don't use Pascal]

Startup

When booting all systems are initialized and checked, then the accelerometer gets calibrated to adjust it with the orientation of the rider. All the periodic background tasks (often referred as "**demons**" in the source code) are initialized to be fired regurarly using the Alarm mechanism provided by Erika3.

During this step it is recommended to sit over the motorbike in your preferred riding position.

Communication with the Android Smartphone

At low level all datas are received using a serial connection with an high baudrate. I've developed a custom protocol which uses "**software endpoints**" in order to address data received from the smartphone. Endpoints are configure using a macro inside `driver/bluetooth_endpoints.c`

Each endpoint has an id number and a fixed size in byte. Each succesful reception is acknoledged back and the helmet can request new data. If the smartphone attempts to send data to a non-empty endpoint an error is sent back the trasmission is automatically retried after few milliseconds. For trasmission, a dynamic linked list of unallocated buffers is used.

Along the source code, a testing **Android application** is provided, in order to trigger demo notifications. The helmet costantly pings the application and shows the connection state in the top-right corner of the screen (a red cross means disconnected, while the bluetooth

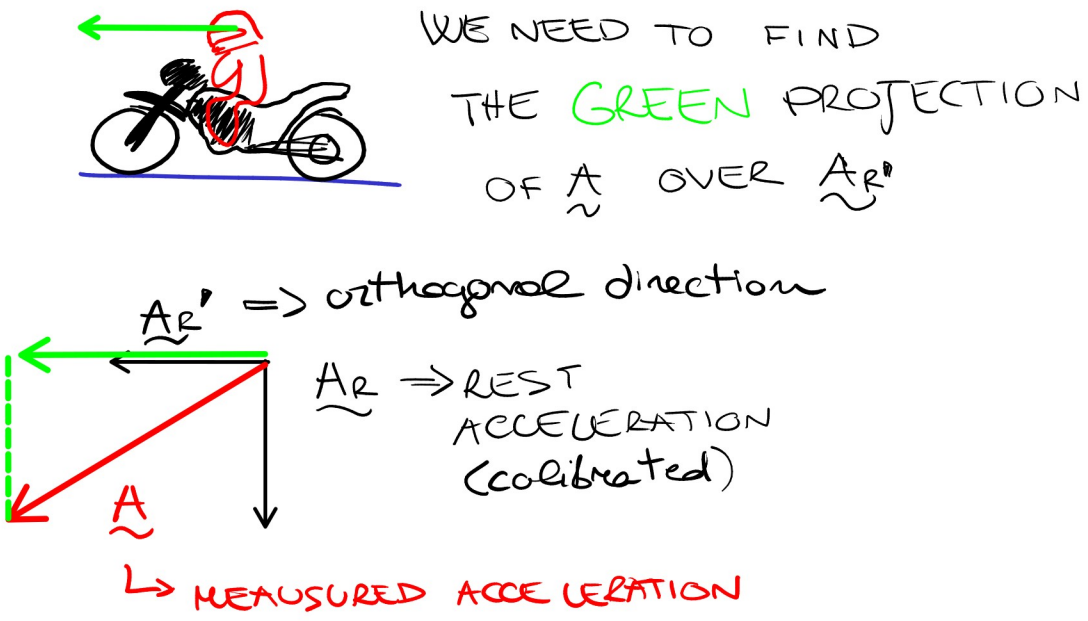
symbol means connected). The app also provide the current time in hours and minutes format, this information is requested to the app every 15 minutes because the internal oscillator of the CortexM4 core has a lot of drift.

Inertial acceleration measurement

The speed data is fed using the gps of the smartphone, however, since gps normally has a high latency the actual speed is computed integrating over short periods the acceleration sensed by the accelerometer.

The computation of the acceleration is straightforward:

1. The rest acceleration **Ar** of the accelerometer is calibrated at startup
2. The total inertial acceleration, **A** is read from the sensor every 100 milliseconds
3. Since we're working on a 2D plane it's easy to obtain a vector orthogonal to **Ar**, which points along the ground and thus giving the projection line to calcute the magnitude of the accelerating or braking force applied to the motorbike.



Using a bit of vector algebra it's easy to obtain **Ar'** from **Ar**: given Ar with cordinates **(x,y)** Ar' is given by **(y, -x)**.

We need to remember that **A**, the total acceleration also countains the contribute of the gravity. Thus, to calculate the real acceleration affecting the motorbike, filtering out gravity, we can just use:

$$effective\ acceleration = (A - \underline{A_r}') \cdot \underline{A_r}' / \|\underline{A_r}'\|$$

then the accelration is ready to be used in the subsequent calculations.

Speed and stopping distance computation

Speed is calculated integrating the acceleration over short time intervals, reversing the definition of acceleration:

$$\Delta\, speed = acceleration \cdot time\, inteval$$

However, in order to reject the drift of the accelometer this formula is used only when the acceleration pass a defined threshold. Then the gps data is ignored because the response would be too slow in order to show the abrupt change in speed of a sudden brake.

On the other hand, the stopping distance is computed using a well known formula, whose derivation is shown, where **v** is the initial speed and **a** the braking force:

$$space\ travelled = v \cdot time + \frac{1}{2} a \cdot time^2$$

since we brake untile a complete stop, as the acceleration is the rate at which the velocity change during time, we know that:

$$time = \frac{v}{a}$$

which in turn leads to

$$space\ travelled = \frac{1}{2} \cdot \frac{v^2}{a}$$

However we must account the **reaction time**, that is, the time that uses the brain to undestand firstly that it has to brake and then to pull the brake lever. Then we have:

$$space\ travelled = \frac{1}{2} \cdot \frac{v^2}{a} + v \cdot reaction\ time$$

The reaction time estimated is fixed and its value is 1.2 seconds. Better to be safe!

I'm considering to add a humidty sensor in order to increase the stopping distance during rainy weather, due to the minor friction forces that can be excert on the asphalt.

Screen drawing optimizations

This screen uses **565 RGB colors**: this way only two bytes are used for each pixels and higher framerates can be achieved (5 bits for red and blue, while 6 bits for green)

In order to obtain a smooth framerate the source code implements the **double buffering** technique. Two copies of the framebuffer (i.e. all the pixels on the screen) are held in memory and at every moment one is used for drawing and the other is transmitted to the screen using the configured **DMA Channel**. High framerates as 220 FPS can be achieved.

In addition, because of the limited **pixel density** of the screen, all the characters are **blended using a 8 bits alpha** channel, leading to a clean and nice representation even for small fonts (12 pixels is the smallest available). For images, a 4 bits alpha channel is used.

The difference between a font drawn without blending and with blending is quite impressive and thus I believe this is one of the most important strategies used to display things nicely over such a tiny screen.

All images and fonts are converted to bitmaps using a custom python script (not included in the source code as is not relevant to the embedded application).

You can find every image or font used in the application inside `ASSETS/IMAGES` and `ASSETS/FONTS`.

Also every drawing routine is custom made to fit the small 0.96 Inch oled screen, all the drawing-related code is inside `DRIVER/GRAPHICS.C`.

The operating system: Erika Enterprise RTOS v3

On top of all the custom drivers is executing a powerful real time operating system which provides all the synchronization mechanisms used to swap data between the main thread and all the interrupts.

Erika ***resources*** are extensively used throughout the whole embedded application.

Views

The helmet as three different views that can be switched using the first button. All the photos shown were shot without the lens, otherwise the image would be a bit messy.

Riding mode

While riding the speed and the stopping distance are constantly displayed. The current time is shown in the top-left corner. A nice bar is colered blue while travelling at costand speed, and i turns green if accelerating or red if braking, providing a visual feedback of the accelerometer reading.

SatNav mode

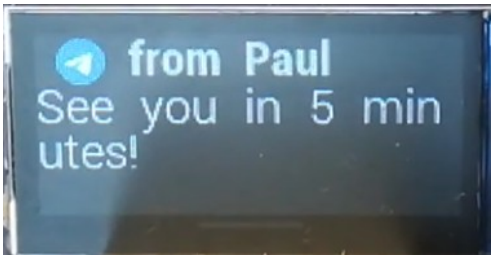
The driving direction are displayed using images and text, along with the external temperature.

Brightness adjustment

Brightness of the screen can be adjusted by hand using the buttons on the side of the helmet.

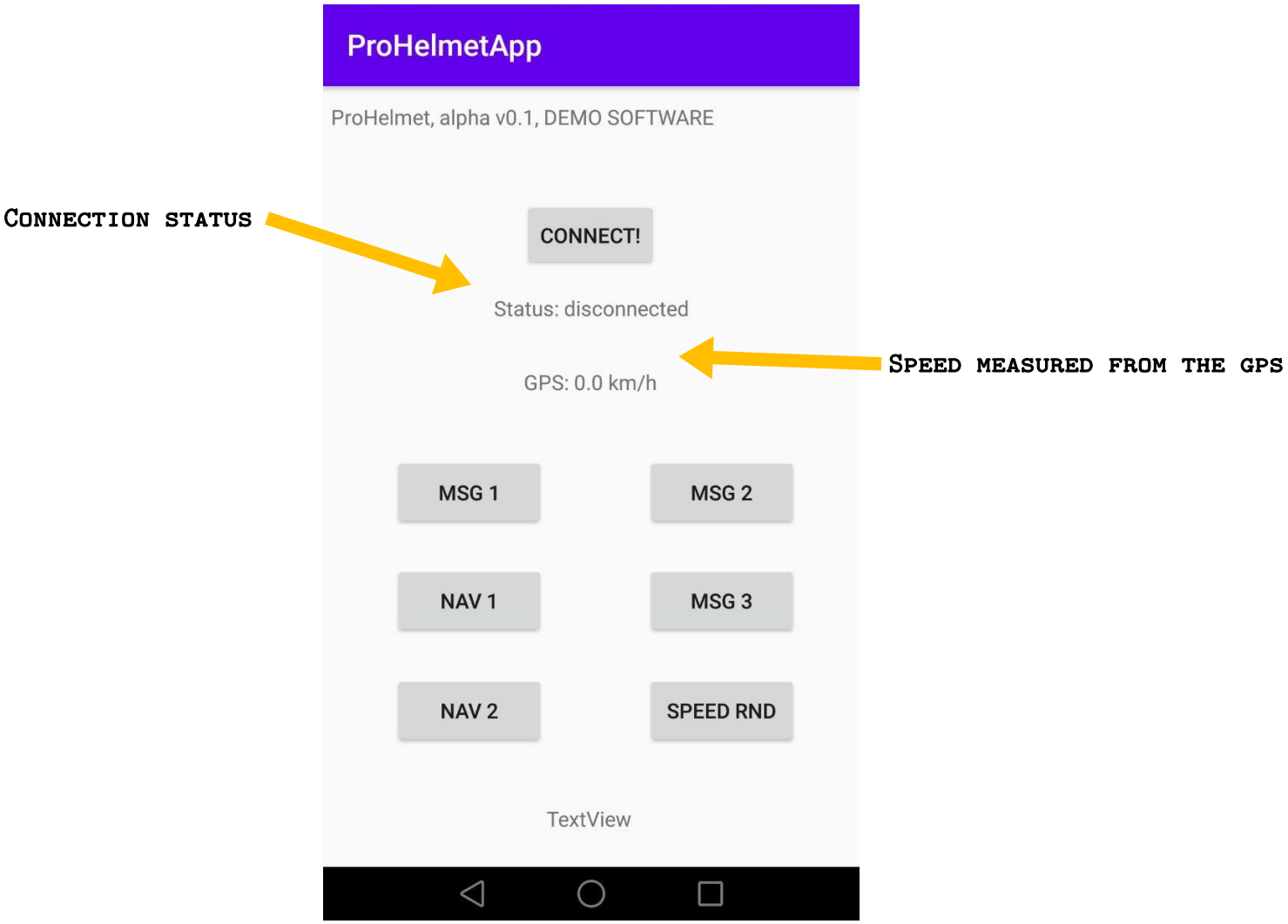
Notifications

On screen push notifications are triggered by the smartphone.



Android Demo Application

In order to show the helmet's capabilities I made an Android Application which can trigger notifications and provide to the helmet some informations like time and current speed, from the internal gps of the smartphone.



Each button can send a different notification or driving direction. Meanwhile, ping echo, **speed** and time are **constantly sent** over bluetooth.

The source code of the android application is available and it is located under `ProHelmet/ProHelmetApp`. The application requires the use of bluetooth devices and gps.

This is only a **brief demonstration** of what could be done with the helmet. The notifications are simulated, although it could be possibile to use actual notifications received by the smartphone.

Moreover, driving direction support could be integrated with services like *Google Maps*, however such integration goes beyond the scope of this project.

As it can be verified looking at the source code, the notifications and the driving directions are sent over bluetooth and thus are **not** pre-programmed in the application running on the helmet's STM32.

Bill of materials

- STM32F429ZI-DISCOVERY
- 0.96 Inch OLED IPS Screen, 160x80 pixels, LCD DRIVER: ST7735.
- HC-05 Bluetooth Serial Module
- ADXL345 Accelerometer
- BMP085 Barometric pressure sensor
- L3G4200D Gyroscope
- 3 Push buttons
- Photoresistor & 30 KhOhm resistance.
- Wires and heat-shrinking tubes