Mobility Smartparking

We started this project with a simple goal: we wanted to measure the incoming and outgoing numbers of a parking lot, and thus inform people about the free and occupied spaces in the lot.

During our work we improved the project with some extra functions, such as tweeting and e-mail sending, so people could get informed easily.

1. First step – Gadgets, parts

To be able to begin working on the project our first step was to get our hand on the necessary parts, which are the following:

● Raspberry Pi 3

https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

● Ultrasonic transducer HC-SR04

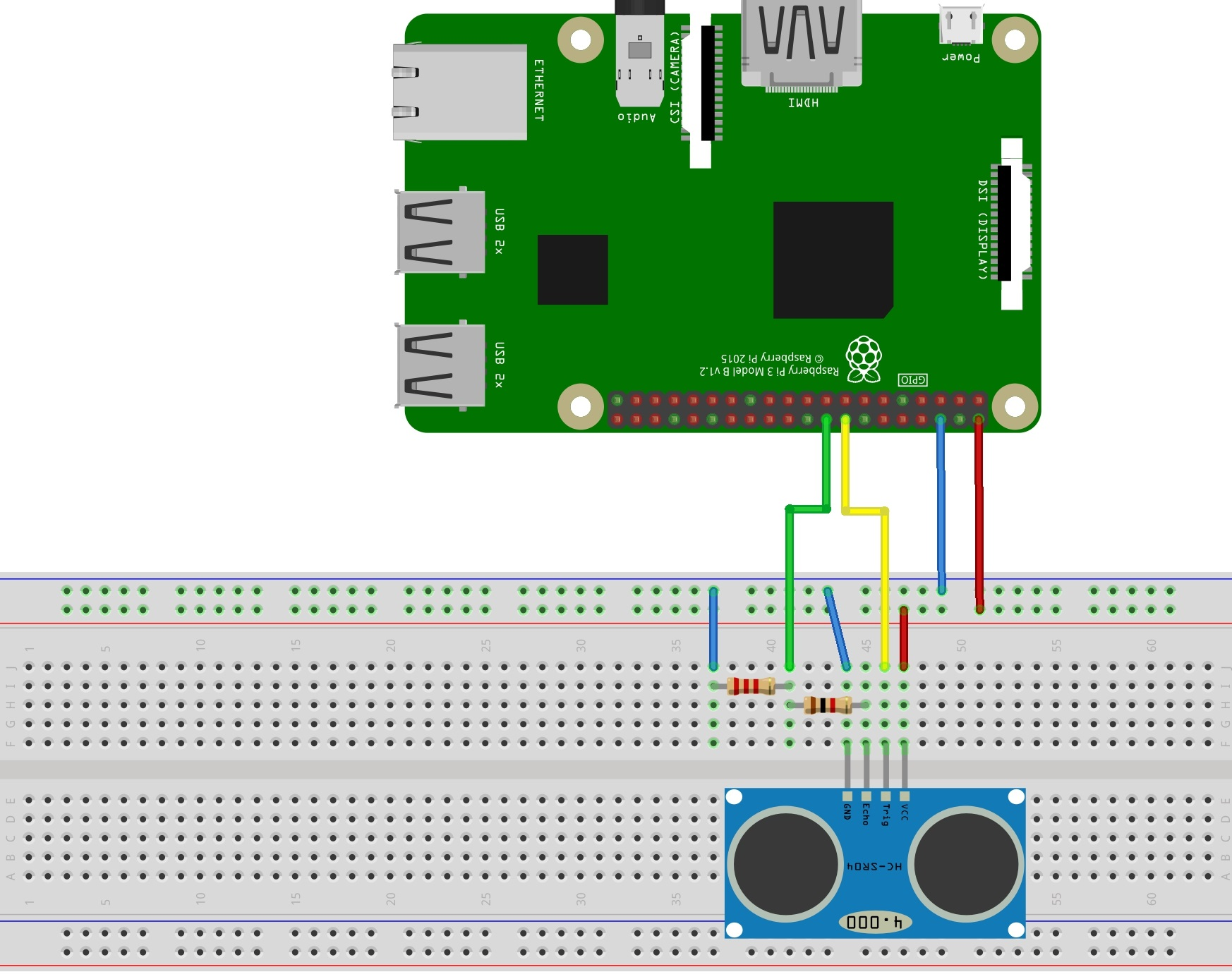
http://hobbielektronikabolt.hu/spd/HCSR04/Ultrahangos-tavolsagmero-HC-SR04

● Dashboard for the sensors, and cables for the linking, with 1000 Ω resistance

● Power supply – Powerbank

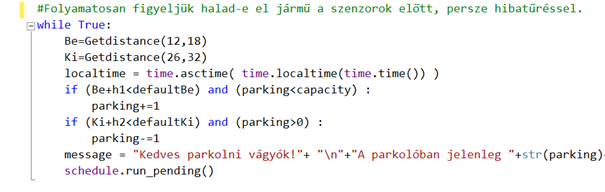
2. Second step – Raspberry Pi and sensors

As our second step we had assembled the hardware part. So we connected the 2 ultrasonic sensors and installed the OS (Raspbian) onto our Raspberry Pi. After that, to test if the sensors were working properly, we wrote a few lines of code in Python 3 and ran some tests.



3. Third step – Writing the main function

In our next step we programmed our basic code. The idea behind it was to detect the incoming and outgoing objects (vehicles). The distance detected when a car would be passing by would be smaller than the original distance measured during the first measurement. Depending on which sensor would detect the object, it would be counted as an outgoing, or incoming car, and thus would either mean a deduction or addition to the occupied spaces.

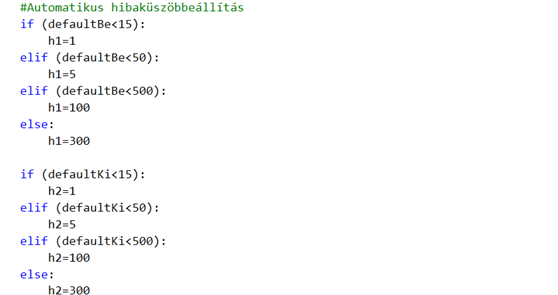


4. Fourth step – Testing

During our work we tested each part of the code, to be able to realize a mistake and to easily check which part of the code had it.

During the testing of our basic code we had to change some parameters. For example the fault tolerance during a place change, and the sleep time of the sensors.

The fault tolerance was first a fix number, but considering that it should be mobile, and so it could be easily set up in any kind of environment we used some different variables in an if condition.

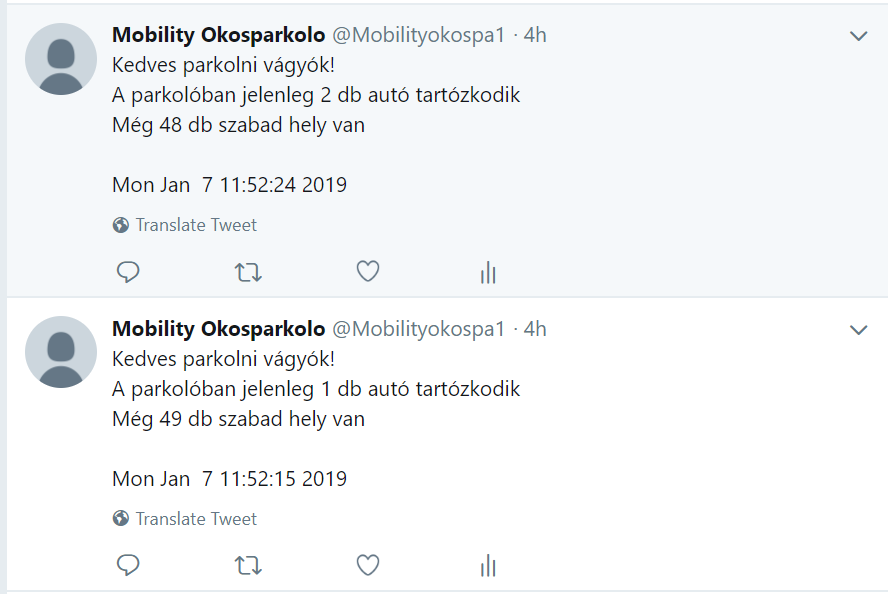


5. Fifth step – Extra functions

In our fifth step we wanted to implement an informing code, which meant that it would occasionally inform people about the parking lots current status.

During this step we first implemented a tweeting and then an e-mail sending part.

Both of these send notifications every 30 minutes, but it can be easily changed.



6. Sixth step – II. Test

In this step we tested the newly implemented elements of the whole code.

In this step we discovered a possible malfunction caused by Twitters rules. Twitter doesn’t allow duplicate posts, so when the number of cars didn’t change after 30 minutes, it would tweet out the same inform. We solved this issue with the use of a time stamp, which also improved the authenticity of the posts.

7. Seventh step – Rehearsal

In our last step we tested the whole system, which included each of the above mentioned parts. This was done in the parking lot of Mobilis with the help of some volunteers. We needed to change some parameters in this case too, so we could count the number of cars without a mistake.

The test was done with the help of 3 people. During this we could determine that the sleep time of the sensors should get a value of 1.5 to perfectly count the cars.

In the following pictures the parking lot, and the set up are to be seen.



