Introduction

In this technical document we will describe the Blinduino project in its wholeness, describing both the electrical part (so Arduino and its shields) and the software one (the frontend app and the backend server).

Blinduino is a system that aids blind and visually impaired users in their daily actions by taking advantage of a simple sensors network which has mainly two functions:

1. Recognize an obstacle in the user’s path and communicate its presence via the Blinduino app so that it can be anticipated and avoided;
2. If requested, once again via the Blinduino app, the OV7670 camera can be used to take a picture of an object to get it identified and described.

Specifically, the obstacle individuation is achieved by using three ultrasonic sensors, which are placed in the following spots, and coupling each one of them with a vibration motor (used to signal the obstacle presence and distance, as it would be done by a car parking sensor):

1. The first one is placed on the head or on the hat, and can be turned on and off using a push button;
2. The second one is placed on the ankle, and can be turned on and off using a push button;
3. The third one is a wearable (can be put on the arm like a watch) which can be moved, and can be turned on and off using the app or using the built-in push button.



Technical Arduino Specification

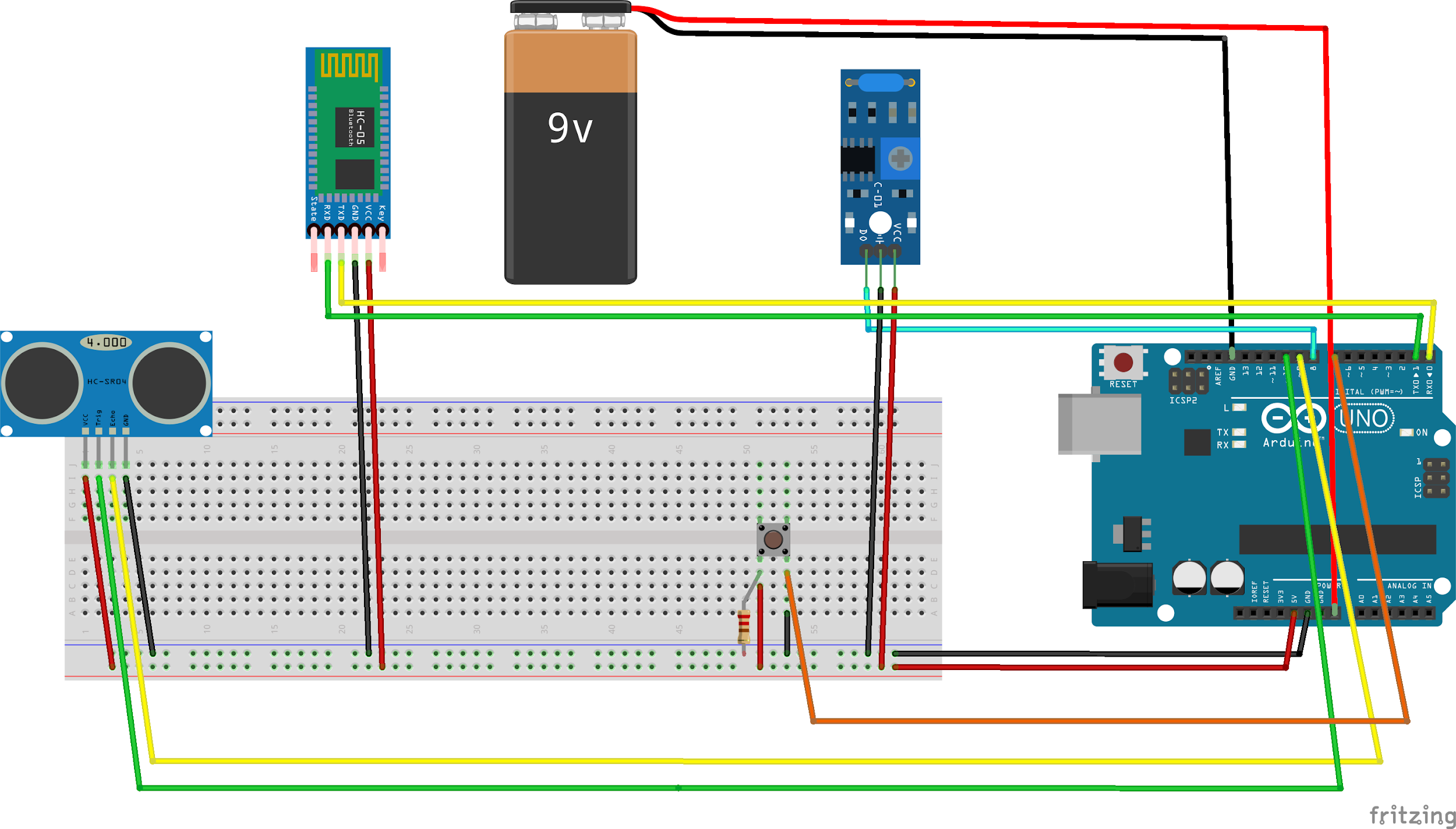
The electrical part of Blinduino is divided in 2 components, a manual sensor and an object recognition camera.

Manual Sensor Composition

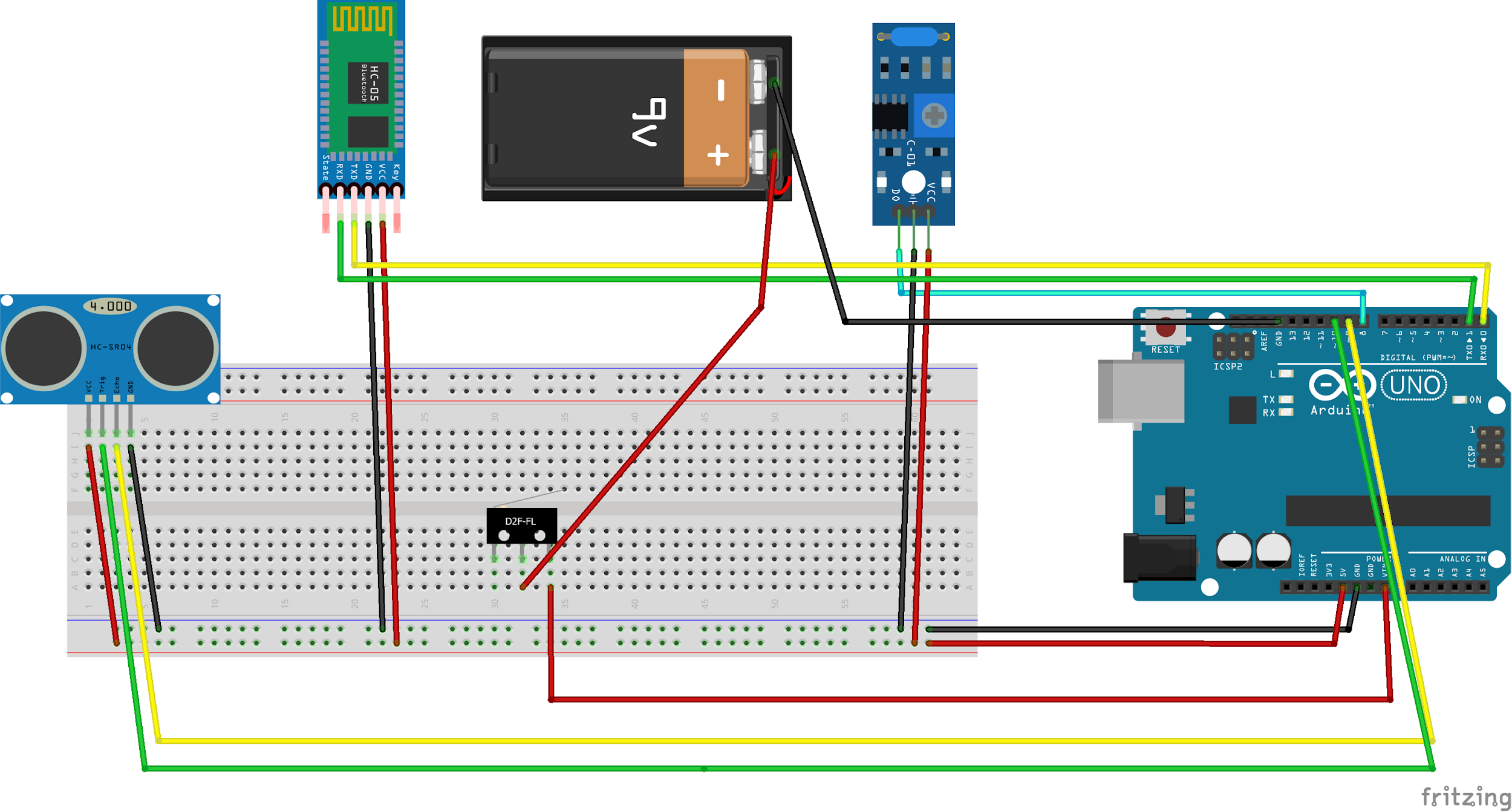
The Manual Bluetooth Distance Meter is linked to a Vibration Motor.

We made two different versions of the same component:

1. The first one has a button (controlled by Arduino) which turns on and off the sensor measurement, the vibration and the bluetooth communication;



1. The second one has a Lever Switch that turns on and off the entire Arduino and everything to it related.



This first component is overall composed of:

* Arduino Uno;
* Ultrasonic Sensor HC-SR04, used to measure distance in a range between 2cm and 4m with an accuracy of 3mm. The sensor module consists of an ultrasonic transmitter, a receiver and the control circuit. The working principle is as follows:
  1. High level signal is sent for 10µs using Trigger;
  2. The module sends eight 40 KHz signals automatically, then detects whether pulse is received or not;
  3. If the signal is received, then it is through high level. The high duration time is the gap between sending and receiving the signal, so the measured distance can be calculated using the formula below.

*Distance= 0.03434\*Time/2*

* Bluetooth Module HC-05, used to connect Arduino with the Smartphone via Bluetooth;
* Push Button or Lever Switch;
* Vibration Motor, which produces vibration based on input;
* Jumpers;
* Resistor 2.2KOhm;
* Breadboard.

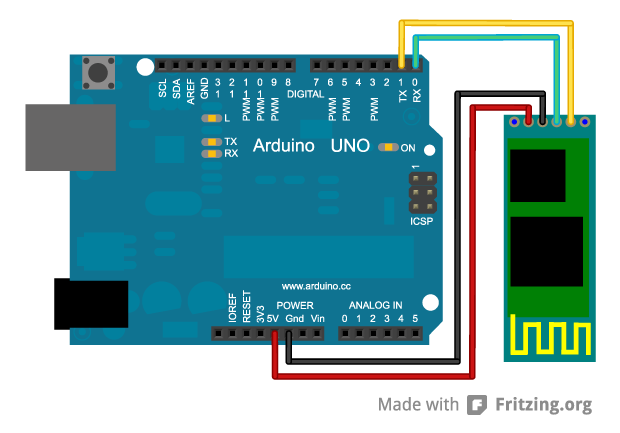
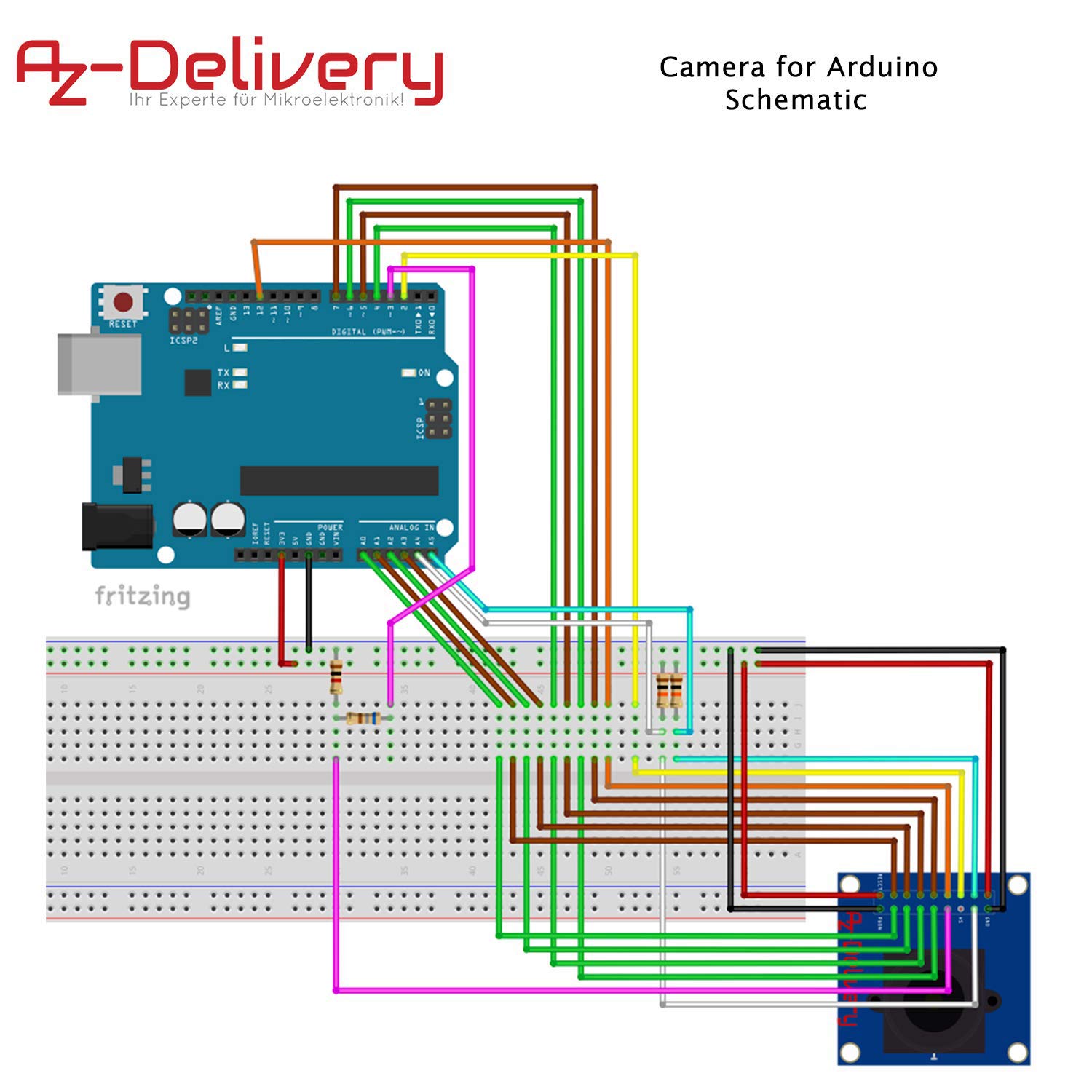
Manual Sensor Behaviour

This component allows a blind user to check for the presence of an obstacle in front of him (via the Ultrasonic Sensor), and in case report its distance using a Vibration Motor set in different vibration frequencies (the time interval between vibrations is smaller if the obstacle is closer, greater otherwise).

The head and ankle sensors are both activated and deactivated using a Push Button or a Switch Lever, while the wrist one can also be toggled via the smartphone app.

All these information (distance, operation status and vibration level) are processed and sent by the HC-05 Bluetooth Module to the smartphone application (frontend).

Object Recognition Camera Composition



This second component is overall composed of:

* Arduino Uno;
* Breadboard;
* Resistors: 10k Ohm, 10k Ohm, 4.7k Ohm, 4.7k Ohm;
* Bluetooth Module HC-05;
* Jumpers (a lot of them);
* OV7670: its image sensor is controlled using Serial Camera Control Bus (SCCB) which is an I2C interface with maximum clock frequency of 400KHz.

Object Recognition Camera Behaviour

The OV7670 Camera is utilized to capture an image and recognize its content, but we will talk more about it in the backend chapter.

Blinduino Frontend Application

The Blinduino Frontend component is a Angular/Ionic Cordova web application which sends the image recognition requests to the backend, but also displays some information gained from the Arduino sensors to be directly communicated to the end user.

Angular and Ionic Cordova

Angular is an open source framework which uses Microsoft’s TypeScript language to create web applications: when coupled with Ionic Cordova, it allows the creation of hybrid web apps, which are available and usable from both desktop and mobile environments, with ease and little struggle.

Frontend application itself

We first generated a tab-based application, which is divided in three sections (or tabs) that represent the main interface components.

All users are helped in the navigation within the app through the use of voice command which explains which tab they are in and what operation they are doing.

The main three tabs are:

* Object recognition
* Obstacle recognition
* About us

In the first tab (Object recognition) we can interact with the phone camera and take a photo, then attend the result of the recognized object.

An example of this tab can be seen below:

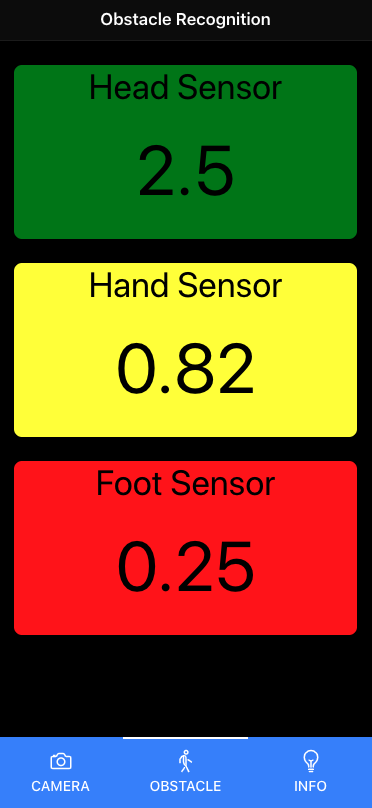


When user clicks on center of the screen, the camera will be activated and once the photo is taken, the request is sent to the backend which will return the recognized object inside the photo and it will be announced vocally.

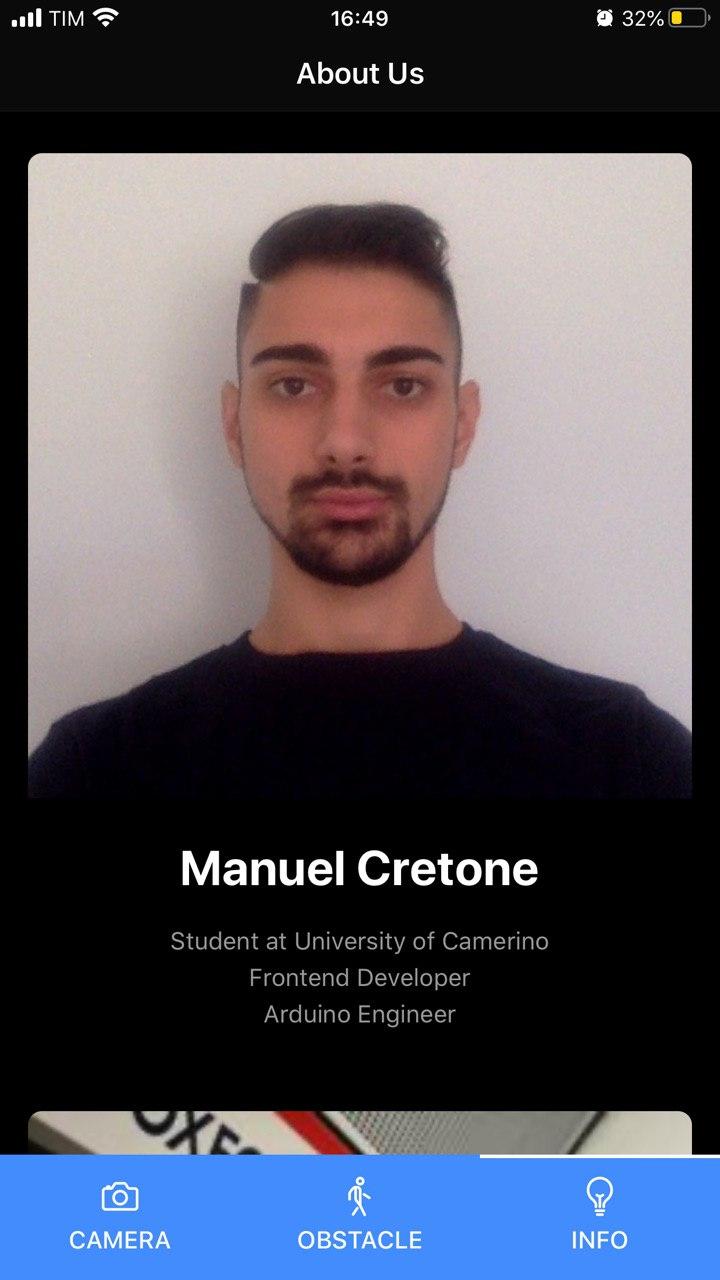
The second tab (Obstacle recognition) shows all the distances detected by the sensors using a color that determines the detected object danger proximity.

Furthermore, in this case users can interact with the card inside the tab, which communicate the distance from the object through voice announcement.

The Object recognition is shown below:



The last tab (About Us) simply shows a brief description of the authors and the roles they had in the project.



Bluetooth Connection

The application also uses a bluetooth connection to communicate with the sensors and, in the case of the sensor placed on the wrist, can also activate and deactivate the sensor itself.

The bluetooth connections allow us to receive from time to time the distances measured by the sensors, and to display/announce them using the text to speech feature inside the application.

Each Arduino/Bluetooth sensor couple have a separate connection, which can be handled by the Bluetooth 4.0+ devices, and data is sent continuously as long as the sensors are active and the bluetooth devices are connected.

Blinduino Backend Server

The Blinduino backend component represents a simple Python 3 app which uses Flask to receive requests and ImageAI/Tensorflow libraries to recognize objects inside images: its whole point is to get, elaborate and return results about photos shot by the arduino camera we described above (the OV7670), which are sent to the backend by the frontend app, that works as a medium in this case.

Python 3

Python is an interpreted, high-level, general-purpose language, which also has some elements of object-oriented and functional programming available: it is a very powerful language which allows the user to execute complicate sequences of action in very few lines of code.

In our case, we wrote the server using the 3.x release of the language, which has lots of differences from the 2.x one, making them 99% incompatible each other: since the 2.x version has been used for years even after the 3.x release and its support has been dropped only a few months ago, we will keep talking about “Python 3” to further specify that the backend is only compatible with this new release.

Flask and Flask-cors

Flask is a microframework which allows to run simple web apps using Python 3 and supports HTTP requests as well as cross-origin handling.

The HTTP requests are different types of calls which can be done to the web app endpoint(s) to send or receive data: in our case, we only used the POST request, which allows to send a call with a body in JSON format and receive a proper response based on the request content.

The cross-origin management is very important since the requests a server can receive are usually blocked if the sender is not trusted (99% of the cases), so it’s important to configure the server to accept all the needed requests and reject the others.

ImageAI and Tensorflow

ImageAI is a common Python library which is used to interact with images and elaborate them using neural networks and machine learning: it uses Tensorflow, an open source platform created by Google exclusively for machine learning, which does almost 90% of the “dirty work”.

The ImageAI library, in our case, is just used to select the neural network type (there are different ones available on the Internet, each one has some pros and some cons), the pre-trained dataset (which we also trained for some bit to obtain better results), and of course the photo to analyze and the number of desired results.

Heroku

It’s the hosting service we used to publish our backend, which supports most of the existing programming languages (including Python 3, of course), HTTP requests and all the other features we needed: it also offers an endless free trial, with decent bandwidth and enough storage space to consistently host our backend and make it usable.

However, we also had some difficulties with it, since tensorflow newer versions are very heavy (so we had to use the older 2.0.1 version, due to the Heroku repos being limited to 500 MB of space) and also very memory-demanding (Heroku offers only 512MB of RAM), but we nonetheless managed to make the backend server operational and only a little bit slow in handling and producing response (this low speed can of course be scaled by using a more powerful hosting service, i.e. a private one).

Backend server itself

After talking about the technologies we used, we will spend a few words on how the server itself works.

As soon as it is booted, it checks if the datasets are present on the homonymous directory, and in case they are not, it downloads them from the internet (they are pretty heavy, and not all of them are mandatory, so we decided to not include them inside the base project).

After that, the backend just waits for a POST request on its root address:

<https://blinduinobackend.herokuapp.com/>

which should contain a JSON in the format below.

*{ “image” = base64image }*

In this case, the base64image received is the one sent by the frontend app, which is then analyzed using the dataset we chose and both the machine learning libraries, and the most likely result is sent back to the frontend as response.

Everything else before and after the steps described above take place in the frontend app, as stated in the previous chapter.

Conclusions

The system we created is an experimental sensors network which, we hope, will help disabled people in their already complicated daily life.

Of course, everything we built up until now is a prototype and can be largely improved, but should work as a proof of concept for what can be done and how to achieve it.

For example, the image recognition neural network can be further trained to obtain a better and more consistent result and its times can be reduced by using a more performant hosting service.

Furthermore, the sensors network could be adapted to be less invasive and to better fit in with the clothes which are commonly used: this would make the “Blinduino experience” a more satisfying and less effort demanding one.

Lastly, using the bluetooth connections, a button can be added on the Arduino that is placed on the hand which activates a secondary feature inside the app meant to find your phone (by reproducing a sound or similar).