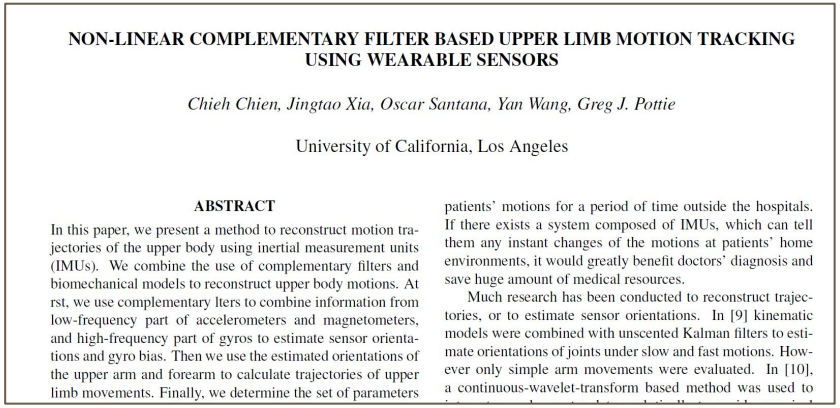
AVANCES HR

Moises Meza

Avances

- Se compró los nuevos materiales para una reestructuración del HR.
- Se leyó artículos acorde a lo acordado.
- Base de datos mongoDB con python

Artículos



NON-LINEAR COMPLEMENTARY FILTER BASED UPPER LIMB MOTION TRACKING USING WEARABLE SENSORS

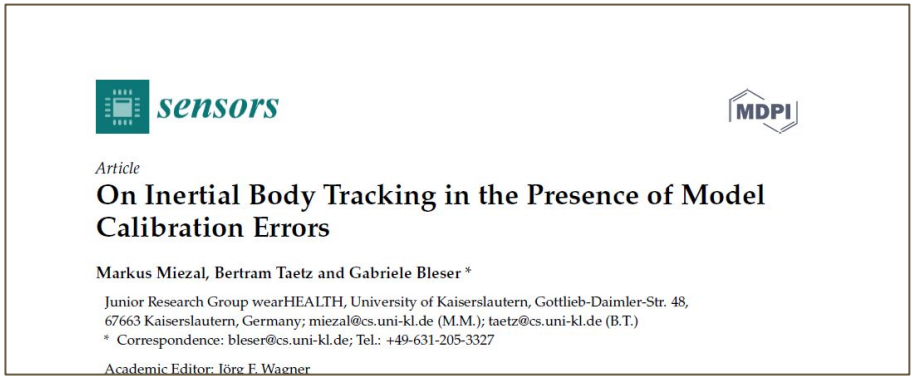
Chieh Chien, Jingtao Xia, Oscar Santana, Yan Wang, Greg J. Pottie

University of California, Los Angeles

ABSTRACT

In this paper, we present a method to reconstruct motion trajectories of the upper body using inertial measurement units (IMUs). We combine the use of complementary filters and biomechanical models to reconstruct upper body motions. At rst, we use complementary lters to combine information from low-frequency part of accelerometers and magnetometers, and high-frequency part of gyros to estimate sensor orientations and gyro bias. Then we use the estimated orientations of the upper arm and forearm to calculate trajectories of upper limb movements. Finally, we determine the set of parameters patients' motions for a period of time outside the hospitals. If there exists a system composed of IMUs, which can tell them any instant changes of the motions at patients' home environments, it would greatly benefit doctors' diagnosis and save huge amount of medical resources.

Much research has been conducted to reconstruct trajectories, or to estimate sensor orientations. In [9] kinematic models were combined with unscented Kalman filters to estimate orientations of joints under slow and fast motions. However only simple arm movements were evaluated. In [10], a continuous-wavelet-transform based method was used to



sensors

MDPI

Article

On Inertial Body Tracking in the Presence of Model Calibration Errors

Markus Miezal, Bertram Taetz and Gabriele Bleser *

Junior Research Group wearHEALTH, University of Kaiserslautern, Gottlieb-Daimler-Str. 48, 67663 Kaiserslautern, Germany; miezal@cs.uni-kl.de (M.M.); taetz@cs.uni-kl.de (B.T.)

* Correspondence: bleser@cs.uni-kl.de; Tel.: +49-631-205-3327

Academic Editor: Jörg F. Wagner

- Usan MPU de 9 grados, incluyen magnetómetro como brújula.
- Usan la PC como adquisición de datos y procesamiento (no es portable).
- Incluyen principios de la teoría de Robótica (cinemática inversa, directa y control de trayectorias).
- Utilizan el filtro complementario.
- Cada MPU lo consideran como un punto de referencia.

Artículos

Universidad del País Vasco — Euskal Herriko Unibertsitatea

Grado en Ingeniería Informática

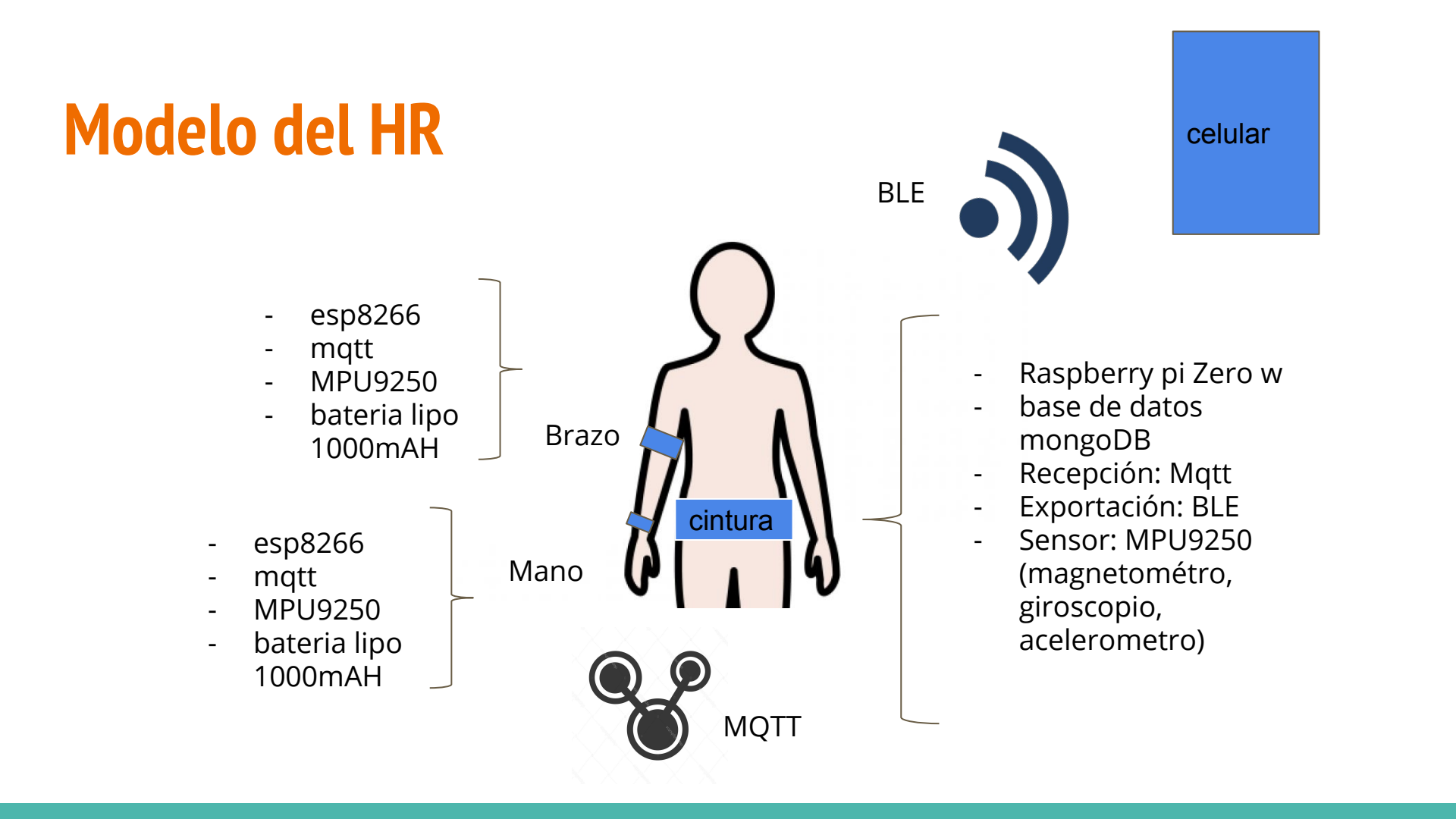
Computación

Proyecto de Fin de Grado

Reconocimiento de posturas mediante Kinect en ROS

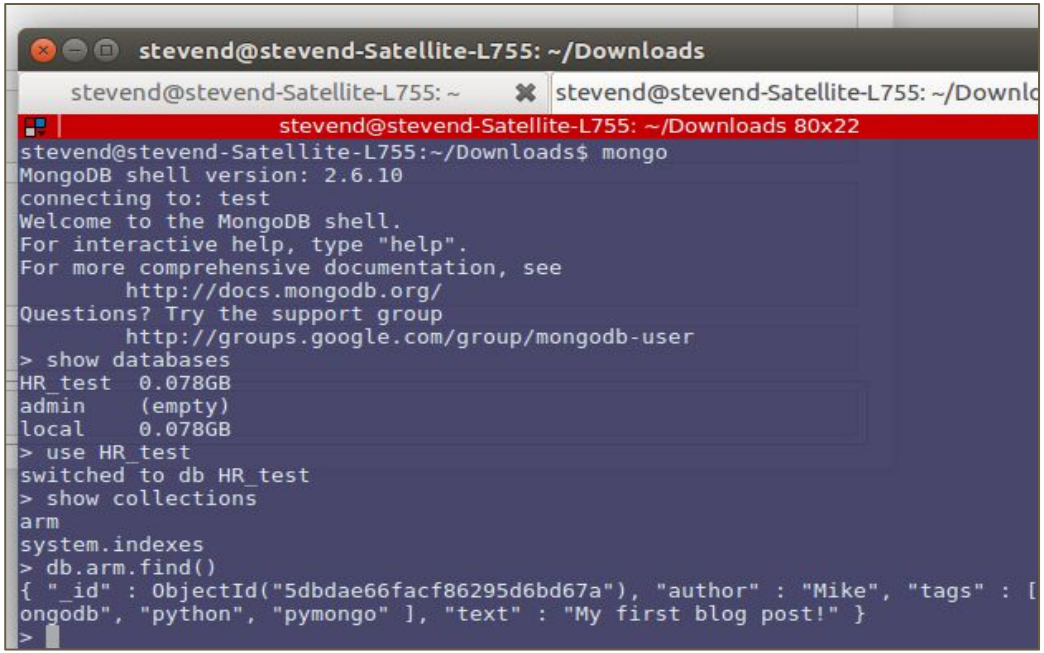
- Obtienen los puntos usando ROS y una kinect 360.
- Usan Gazebo para la visualización de los puntos y crean un nodo para obtener los puntos de las articulaciones.
- Utilizan WEKA como software de análisis y le aplican técnicas de machine learning (recomiendan usar RANDOM FOREST).
- Utilizan C++ como lenguaje de análisis.

Modelo del HR



Base de datos

```
In [2]: from pymongo import MongoClient

In [3]: client = MongoClient()
        client = MongoClient('localhost', 27017)

In [7]: db = client.HR_test
        collection = db.arm

In [8]:  post = {"author": "Mike",
                 "text": "My first blog post!",
                 "tags": ["mongodb", "python", "pymongo"]
                }

In [9]: post_id = collection.insert_one(post).inserted_id

In [ ]:
```



AVANCES HR

Moises Meza

Avances

- Se realizó una mini capacitación sobre Internet of Things, node-red, mqtt.
- Se coordino las compras restantes.
- Se validó el nuevo modelo de hardware
- Se instaló ROS es el raspberry pi zero

Nuevo Modelo

Arduino IDE

GetData2 | Arduino 1.8.10

File Edit Sketch Tools Help

GetData2

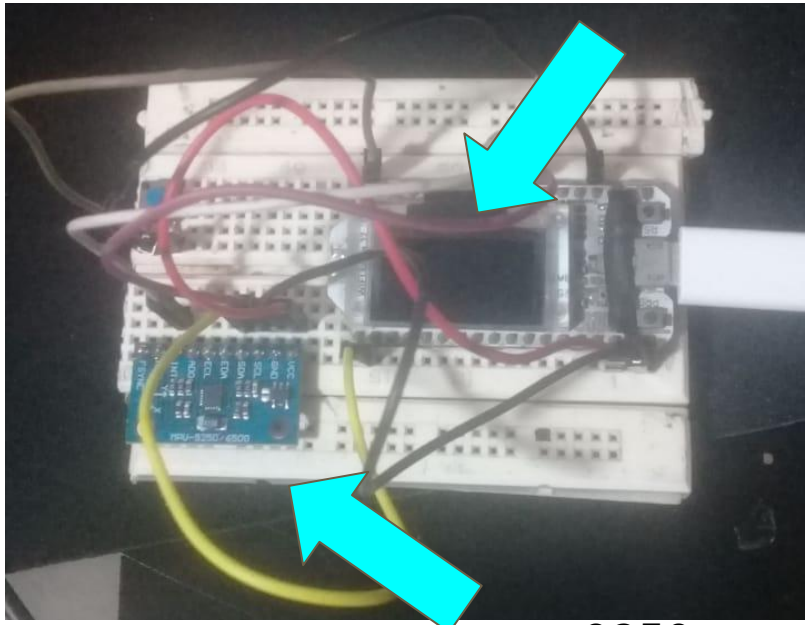
```
#include <MPU9250_asukiaaa.h>

#ifdef _ESP32_HAL_I2C_H_
#define SDA_PIN 21
#define SCL_PIN 22
#endif

MPU9250_asukiaaa mySensor;
float aX, aY, aZ, aSqrt, gX, gY, gZ, mDirection, mX, mY, mZ;
int precBtn1 = HIGH;

void setup() {
  Serial.begin(115200);
  while(!Serial);
  pinMode(16, INPUT_PULLUP);
  //Serial.println("started");


#ifdef _ESP32_HAL_I2C_H_ // For ESP32
  Wire.begin(SDA_PIN, SCL_PIN);
  mySensor.setWire(&Wire);
#endif

  mySensor.beginAccel();
  mySensor.beginGyro();
  mySensor.beginMag();
```

Done uploading.

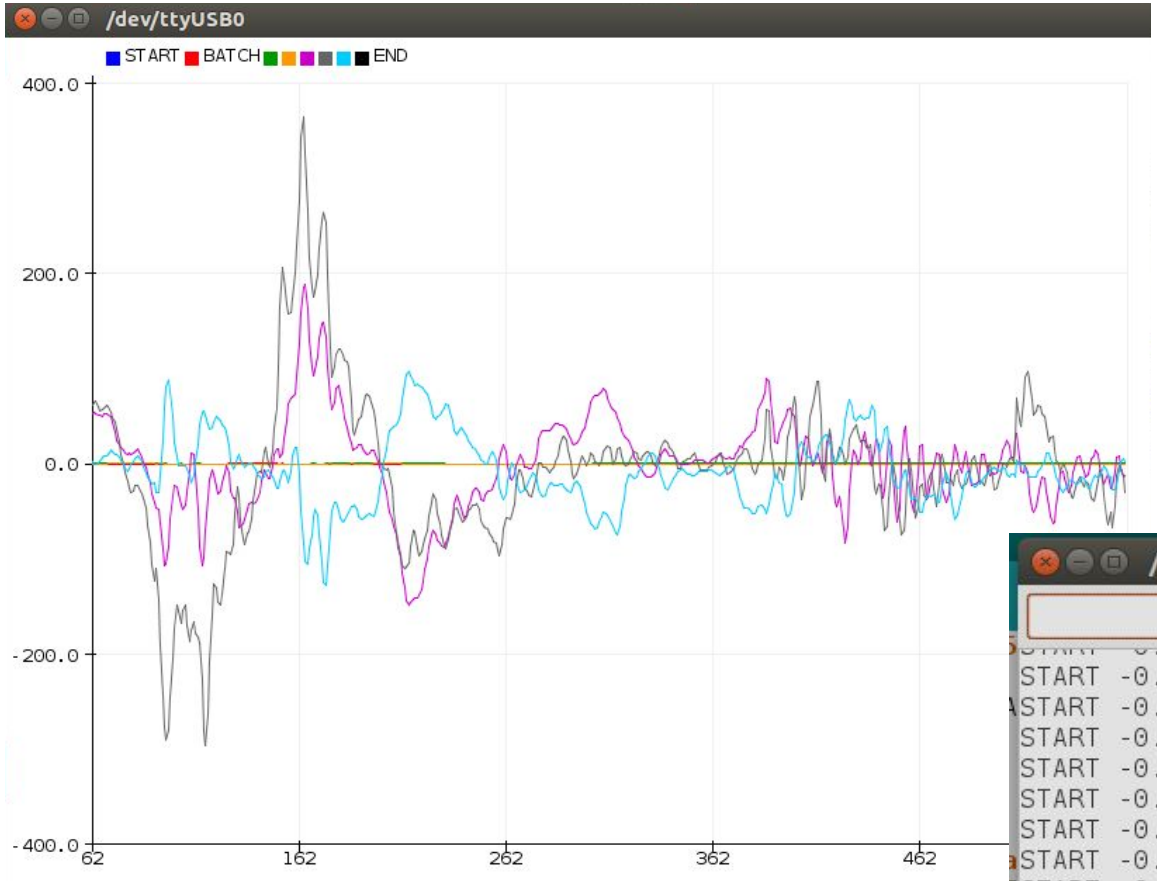
```
Leaving...
Hard resetting via RTS pin...
```

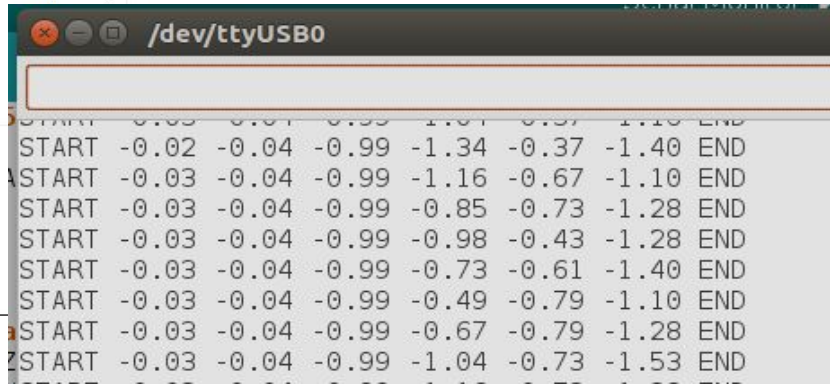
51 ESP32 Dev Module on /dev/ttyUSB0

esp32



mpu9250


/dev/ttyUSB0
START BATCH END
400.0
200.0
0.0
-200.0
-400.0
62
162
262
362
462


/dev/ttyUSB0
START -0.02 -0.04 -0.99 -1.34 -0.37 -1.40 END
START -0.03 -0.04 -0.99 -1.16 -0.67 -1.10 END
START -0.03 -0.04 -0.99 -0.85 -0.73 -1.28 END
START -0.03 -0.04 -0.99 -0.98 -0.43 -1.28 END
START -0.03 -0.04 -0.99 -0.73 -0.61 -1.40 END
START -0.03 -0.04 -0.99 -0.49 -0.79 -1.10 END
START -0.03 -0.04 -0.99 -0.67 -0.79 -1.28 END
START -0.03 -0.04 -0.99 -1.04 -0.73 -1.53 END

Modelo de ML

- Se probó el modelo de machine learning SVM.

What is pyGARL?

pyGARL stands for *Python Gesture Analysis and Recognition Library* and, as you may expect, it's used to build gesture recognition systems. I decided to build it because, after the success of my previous project (the Gesture Keyboard), I felt that a general purpose gesture recognition library would have been useful for a lot of people.

Installation

The installation is pretty straightforward:

```
pip install pygarl
```

NOTE TO WINDOWS USERS: the package requires scikit-learn and scipy to work. The easy way to install them is by using Anaconda (https://www.anaconda.com/download/).

Documentation

All the documentation can be found in the Wiki: https://github.com/federico-terzi/pygarl/wiki

License

This library is distributed using the Apache License 2.0, so you can use it in your commercial projects for free, but please mention both me and the library.

test_ml.py — ~/Documents/lid_biomedica/health_recover/scripts — Atom

test_ml.py

```
1  from __future__ import print_function
2  from pygarl.base import CallbackManager
3  from pygarl.classifiers import SVMClassifier
4  from pygarl.mocks import VerboseMiddleware
5  from pygarl.data_readers import SerialDataReader
6  from pygarl.predictors import ClassifierPredictor
7  from pygarl.sample_managers import DiscreteSampleManager
8
9  MODEL_PATH="stevend/model.svm"
10 PORT="/dev/ttyUSB0"
11
12 def receive_character(number):
13     print(number)
14
15 sdr = SerialDataReader(PORT, expected_axis=6, verbose=False)
16 manager = DiscreteSampleManager()
17 sdr.attach_manager(manager)
18 classifier = SVMClassifier(model_path=MODEL_PATH)
19 classifier.load()
20 classifier.print_info()
21 predictor = ClassifierPredictor(classifier)
22 manager.attach_receiver(predictor)
23 callback_mg = CallbackManager(verbose=True)
24 predictor.attach_callback_manager(callback_mg)
25
26 # Bind all the numbers
27 for c in ["1", "2", "3", "4"]:
28     callback_mg.attach_callback(c, receive_character)
29
30 # Open the serial connection
31 sdr.open()
32 print("Opened!")
33 # Start the main loop
34 sdr.mainloop()
35
```

~/Documents/lid_biomedica/health_recover/scripts/test_ml.py 32:17

AVANCES HR

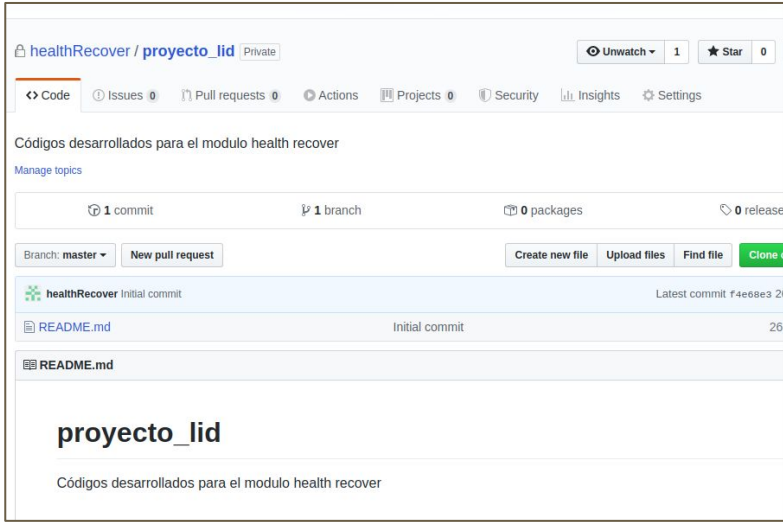
Moises Meza

Avances

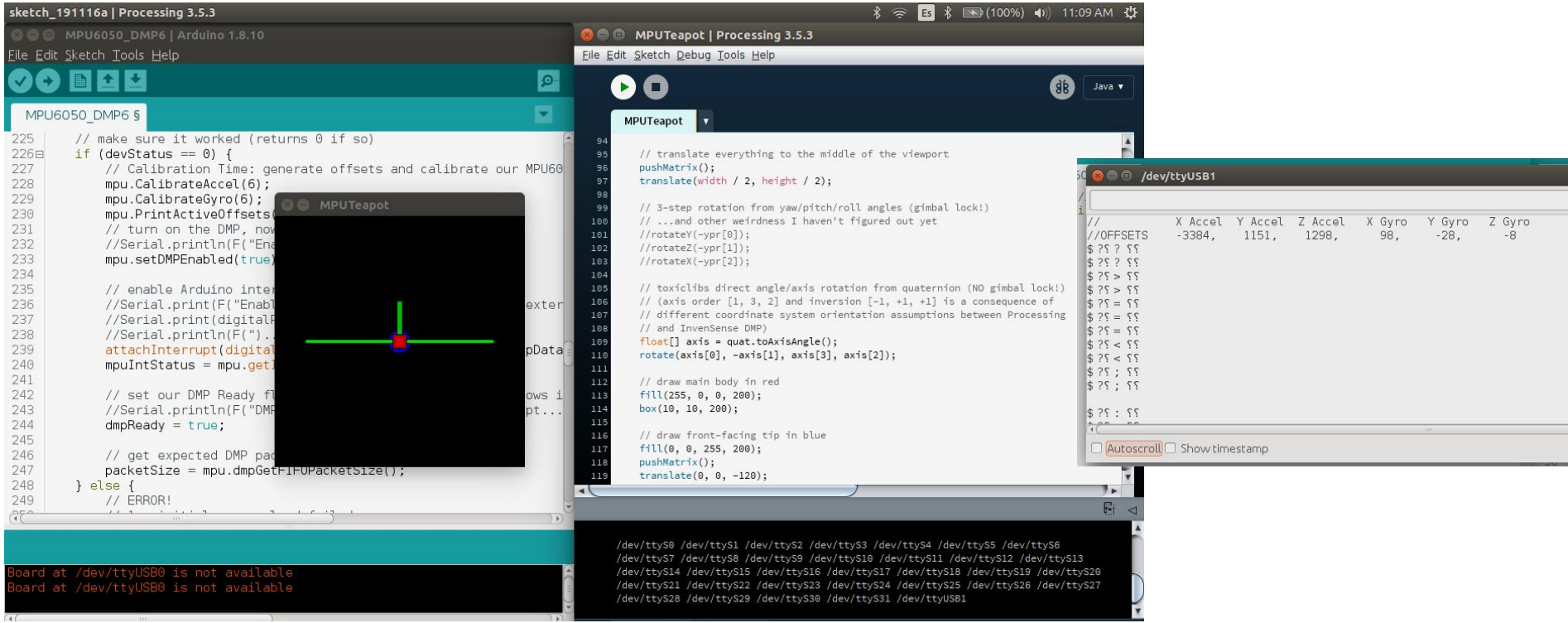
- Se realizó la compra final de componentes.
- Se creó la cuenta en github:

https://github.com/healthRecover/proyecto_lid

- Se avanzó con la interacción del esp32
- Enlace de la raspberry pi zero con mpu9250



Interacción del HR

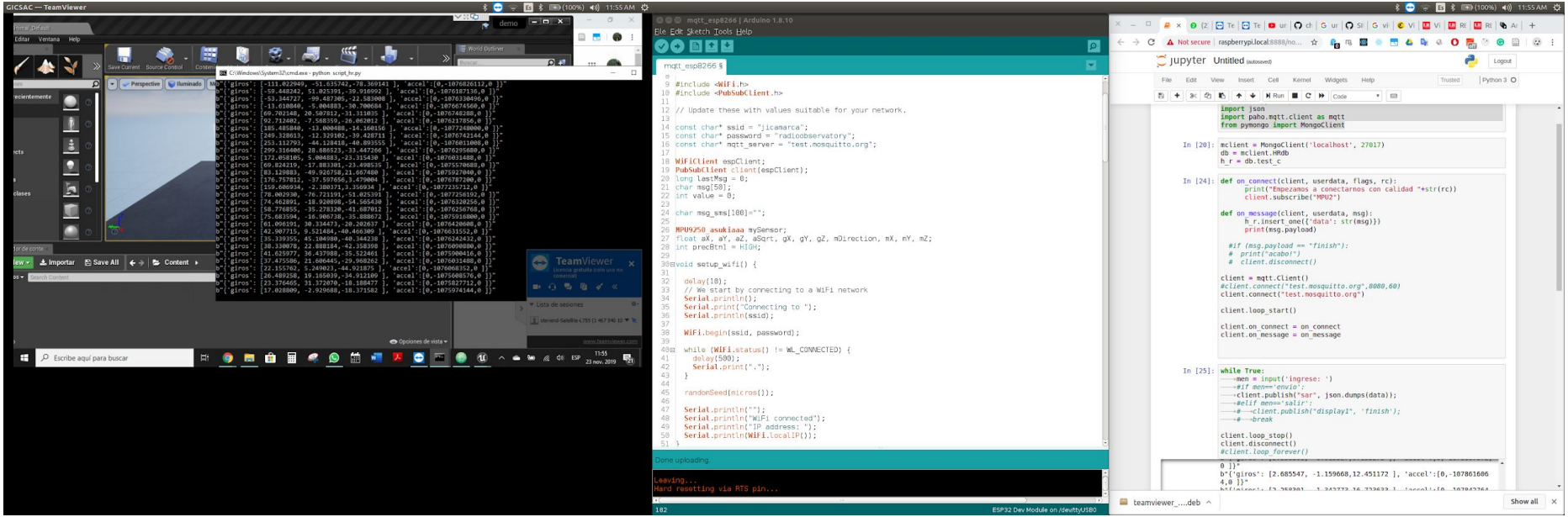


AVANCES HR

Moises Meza

Avances

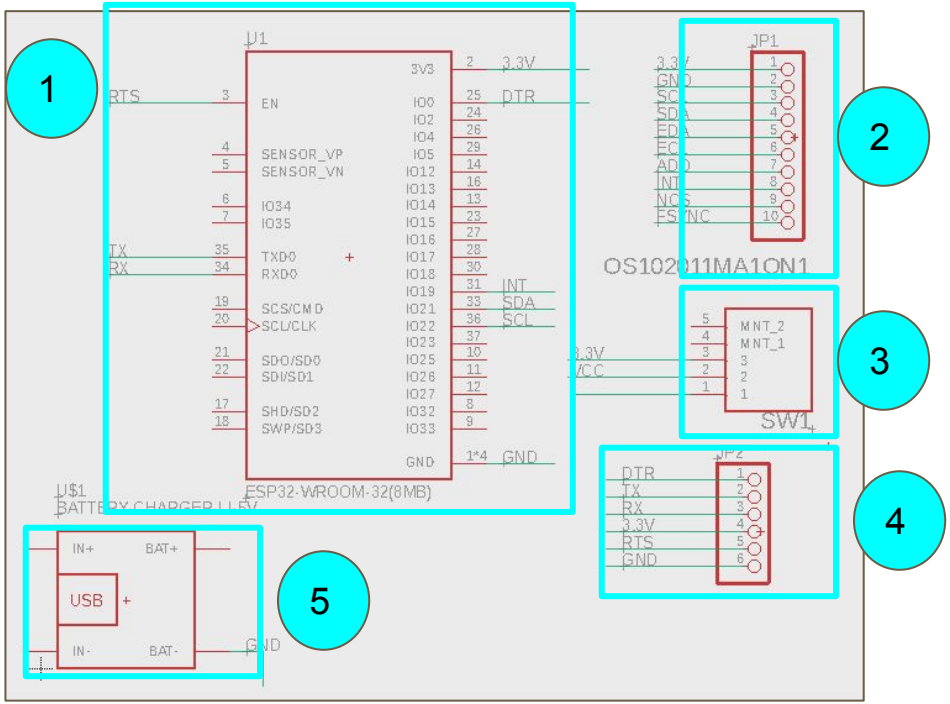
- Se realizó algoritmos de envío de data del esp32 y el mpu9250 usando mqtt.
- Se conectó con el unreal engine, sin embargo se necesita hacer mas pruebas.


GICSAC — TeamViewer
C:\Windows\System32\cmd.exe - python script_hr.py
mqtt_esp8266 | Arduino 1.8.10
File Edit Sketch Tools Help
mqtt_esp8266 §
#include <WiFi.h>
#include <PubSubClient.h>
// Update these with values suitable for your network.
const char* ssid = "jicamarca";
const char* password = "radioobservatory";
const char* mqtt_server = "test.mosquitto.org";
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;
char msg_sms[100]="";
MPU9250_asukiaaa mySensor;
float aX, aY, aZ, aSqrt, gX, gY, gZ, mDirection, mX, mY, mZ;
int precBtn1 = HIGH;
void setup_wifi() {
delay(10);
// We start by connecting to a WiFi network
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
randomSeed(micros());
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}
Done uploading.
Leaving...
Hard resetting via RTS pin...
ESP32 Dev Module on /dev/ttyUSB0
jupyter Untitled (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Not secure raspberrypi.local:8888/no...
import json
import paho.mqtt.client as mqtt
from pymongo import MongoClient
In [20]:
mclient = MongoClient('localhost', 27017)
db = mclient.HRdb
h_r = db.test_c
In [24]:
def on_connect(client, userdata, flags, rc):
print("Empezamos a conectarnos con calidad "+str(rc))
client.subscribe("MPU2")
def on_message(client, userdata, msg):
h_r.insert_one({'data': str(msg)})
print(msg.payload)
#if (msg.payload == "finish"):
# print("acabo!")
# client.disconnect()
client = mqtt.Client()
#client.connect("test.mosquitto.org",8080,60)
client.connect("test.mosquitto.org")
client.loop_start()
client.on_connect = on_connect
client.on_message = on_message
In [25]:
while True:
men = input('ingrese: ')
#if men=='envio':
client.publish("sar", json.dumps(data));
#elif men=='salir':
# client.publish("display1", 'finish');
# break
client.loop_stop()
client.disconnect()
#client.loop_forever()
teamviewer_....deb
Show all

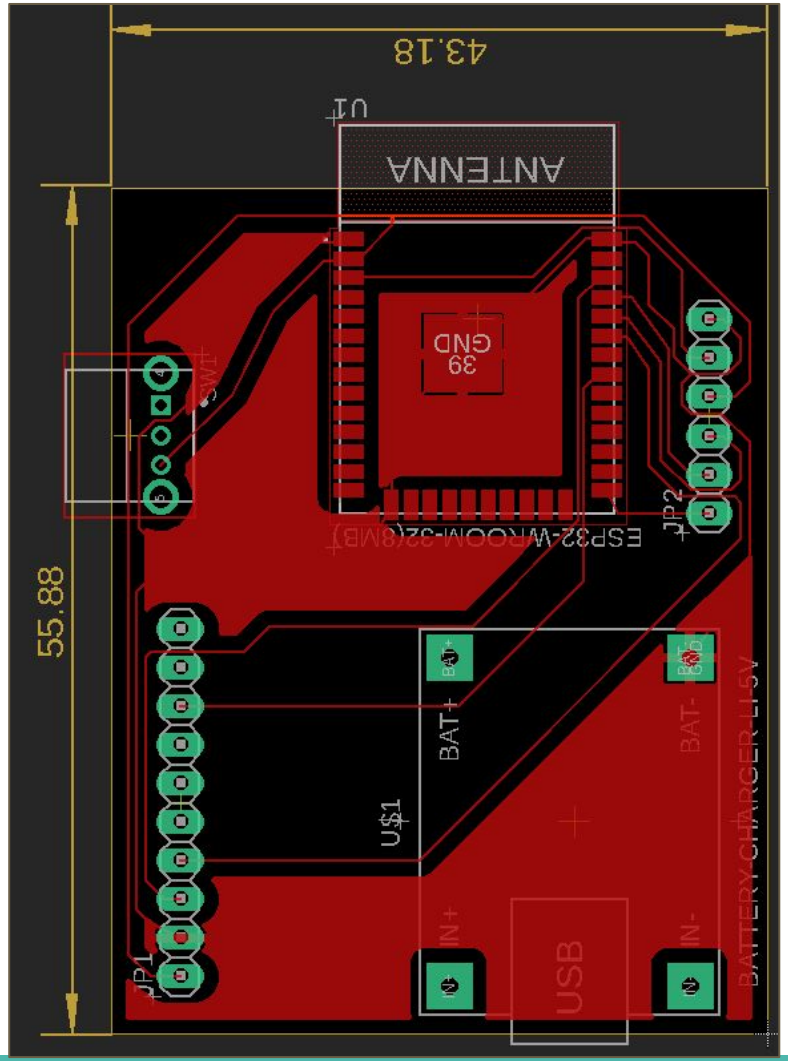
AVANCES HR

Moises Meza

Desarrollo de la segunda versión del dispositivo



1: Esp32 modelo: wroom
2: MPU9250 (solo espadines)
3: micro slider switch 4mm
4: Pines para carga del firmware
5: Módulo cargador de batería


43.18
U1
ANTENNA
GND
39
ESP32-WROOM-32(MB)
JP2
55.88
BAT+
BAT-
U$1
JP1
USB
IN+
IN-
BATTERY CHARGER LI-5V

35 mm

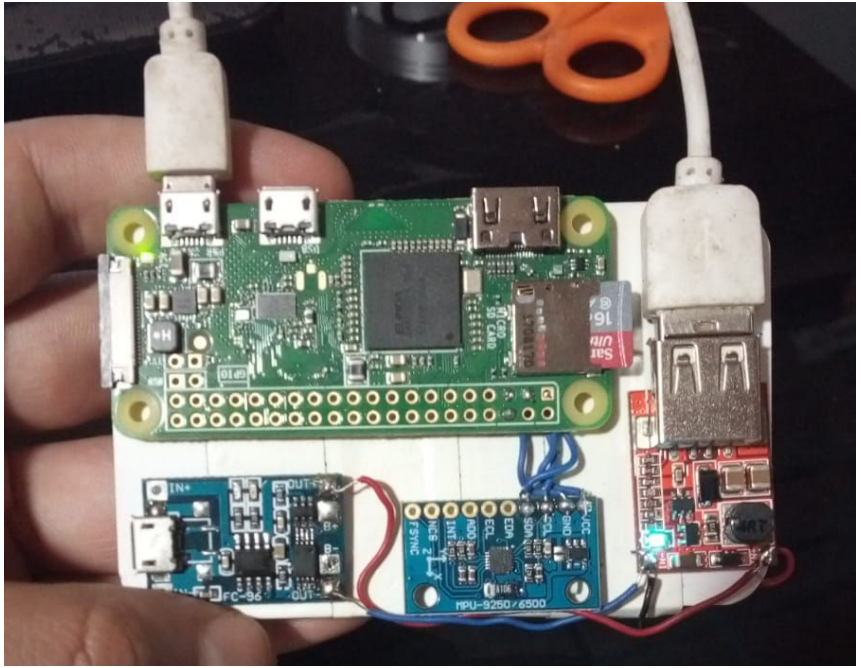

- CL
52345
+ 3.7V 1000mA
ZIFCONN China

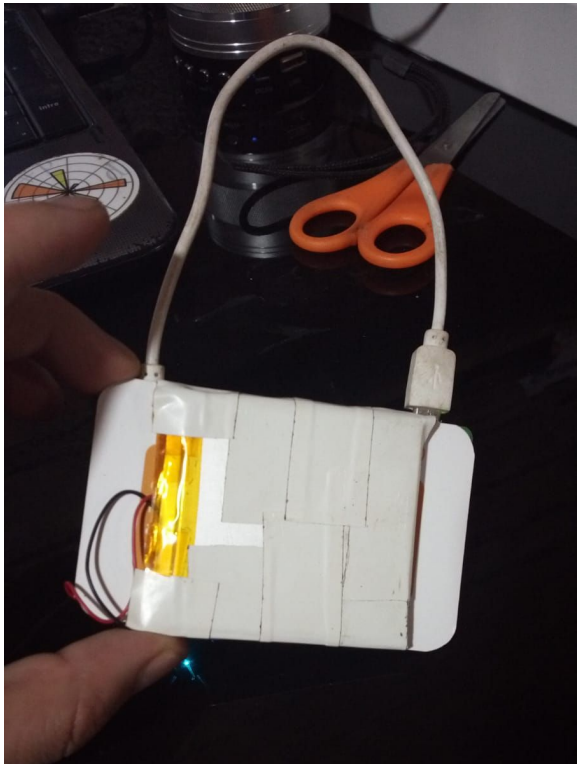
52 mm

AVANCES HR

Moises Meza

Desarrollo del prototipo de HR - cintura





```
(' ax = ', 0.068)
(' ay = ', 0.072)
(' az = ', 1.033)
(' gx = ', -0.732)
(' gy = ', 0.671)
(' gz = ', -0.587)
(' mx = ', 39.886)
(' my = ', -7.656)
(' mz = ', -67.212)

(' ax = ', 0.071)
(' ay = ', 0.07)
(' az = ', 1.031)
(' gx = ', -0.61)
(' gy = ', 0.488)
(' gz = ', -0.435)
(' mx = ', 37.953)
(' my = ', -6.766)
(' mz = ', -66.699)

(' ax = ', 0.068)
(' ay = ', 0.072)
(' az = ', 1.032)
(' gx = ', -0.725)
(' gy = ', 0.565)
(' gz = ', -0.641)
```

pi@raspberrypi: ~/Docum