SAURUSS

Smart
Autonomous
UAV
Recognizer for
Universal
Surveillance
System

What is SAURUSS?

- SAURUSS is an innovative surveillance system that use drones to search and spot intruders
- It's completely autonomous!
- It will find threats, capture footage and notify you, keeping you safe... ...and informed

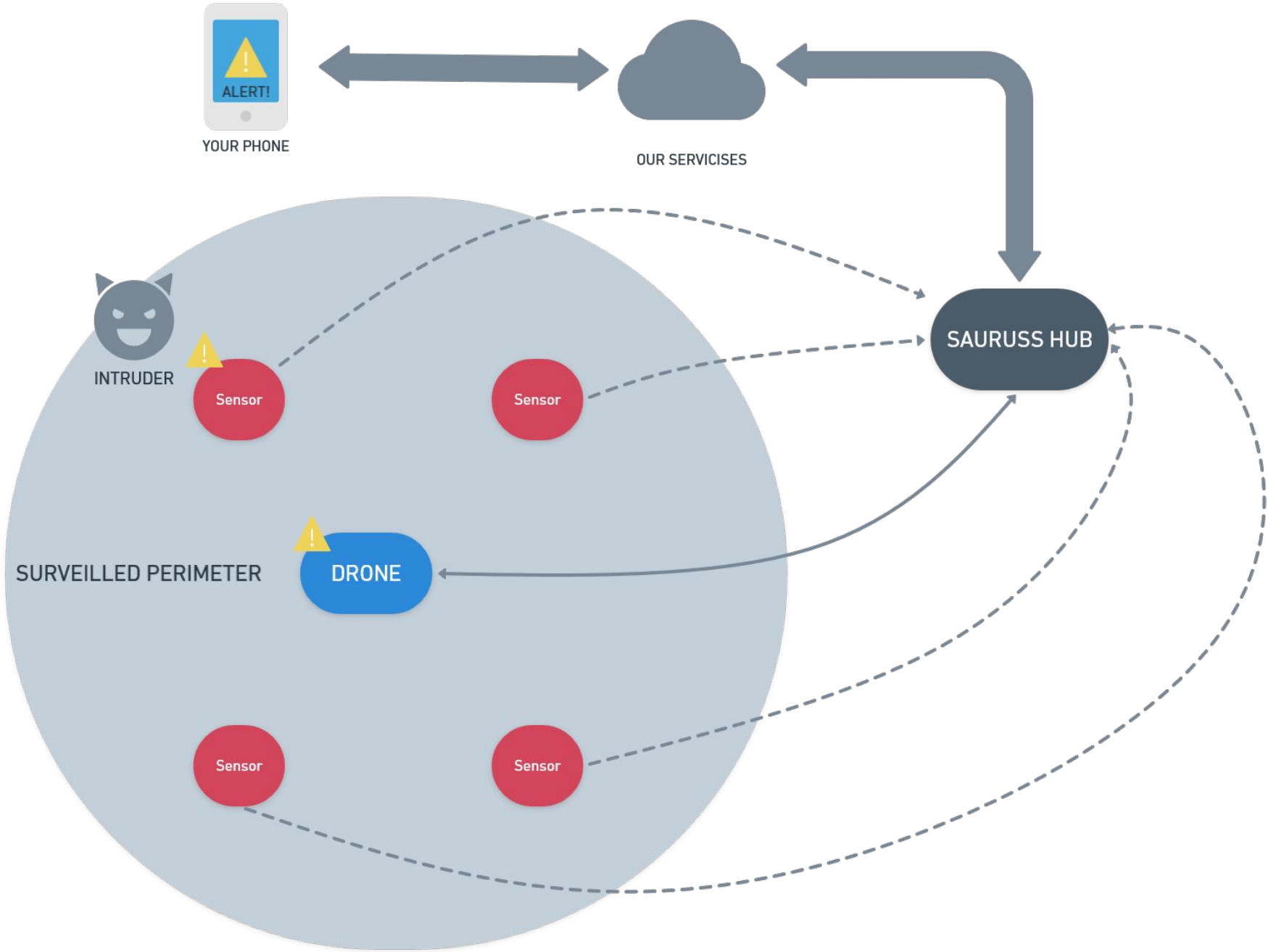
Our System

SAURUSS comes as kit, easy to install and ready to go:

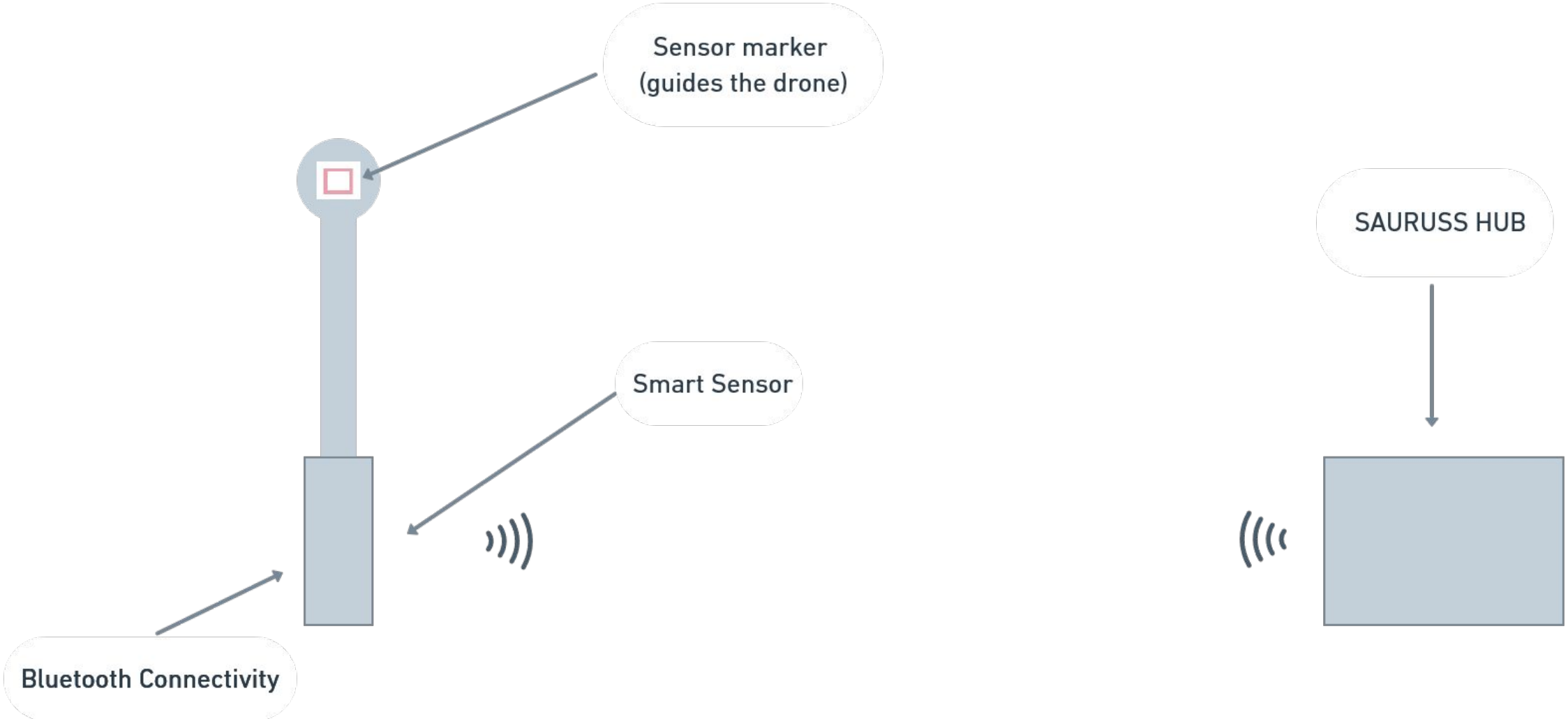
- our drone, with his home
- our sensors (you can use a custom number of them)
- our SAURUSS Hub
- our companion phone app: SaurAPP

How it Works?

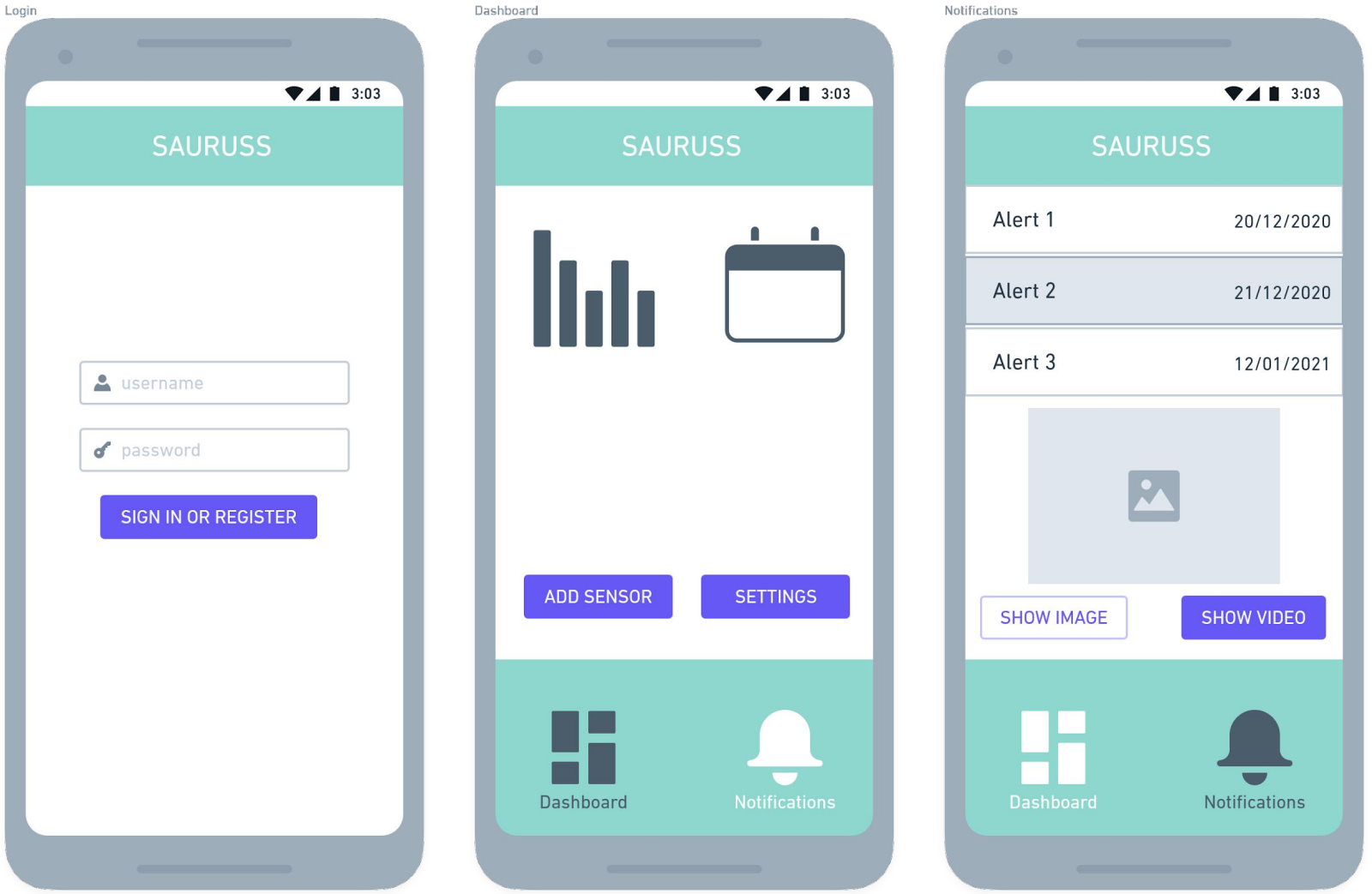
- If our sensors sense an intruder inside your surveilled perimeter, our drone will go towards it.
- Our smart hub will tell if there is someone dangerous and notify you,through our app, with a pic and video proof! 📸
- Say cheese, intruder!



OUR SENSORS AND SAURUSS HUB



Our companion app: SaurApp



- **Register and Login**
- **Dashboard**: info and scheduling for ON/OFF of SAURUSS Sensors
- **Notifications**: see new and past alerts, get photos and video proofs
- **Easy setup**: scan the QR code on our sensors and you're done!

Why SAURUSS ?

Easy Setup

Fully Autonomous

Flexible

private, industrial, military,...

Mobile

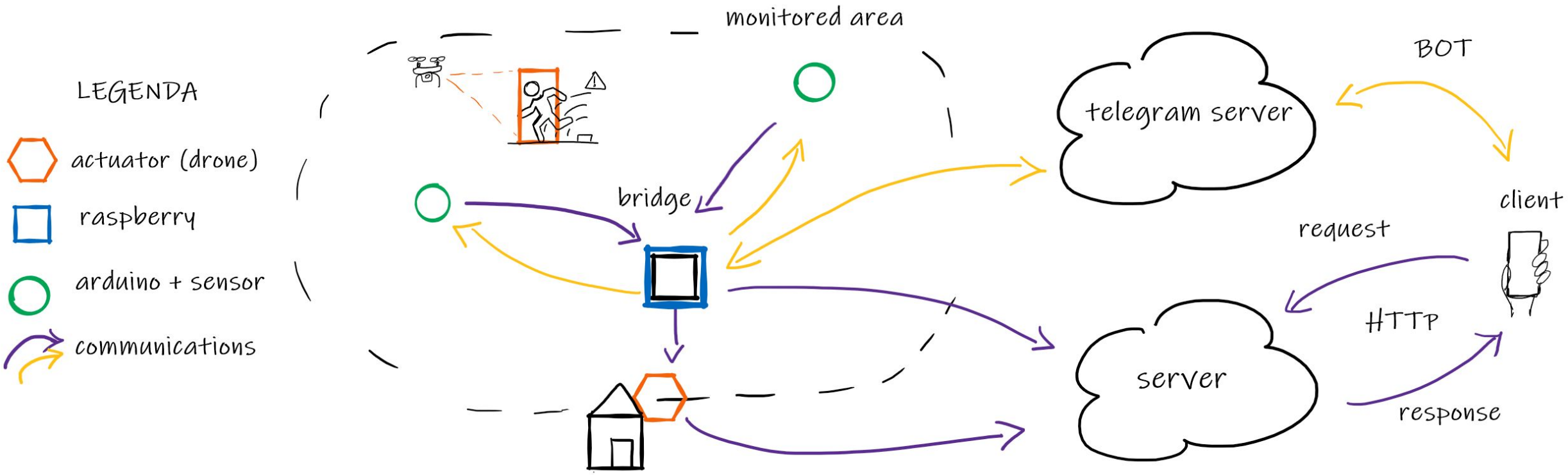
HW & SW architecture 

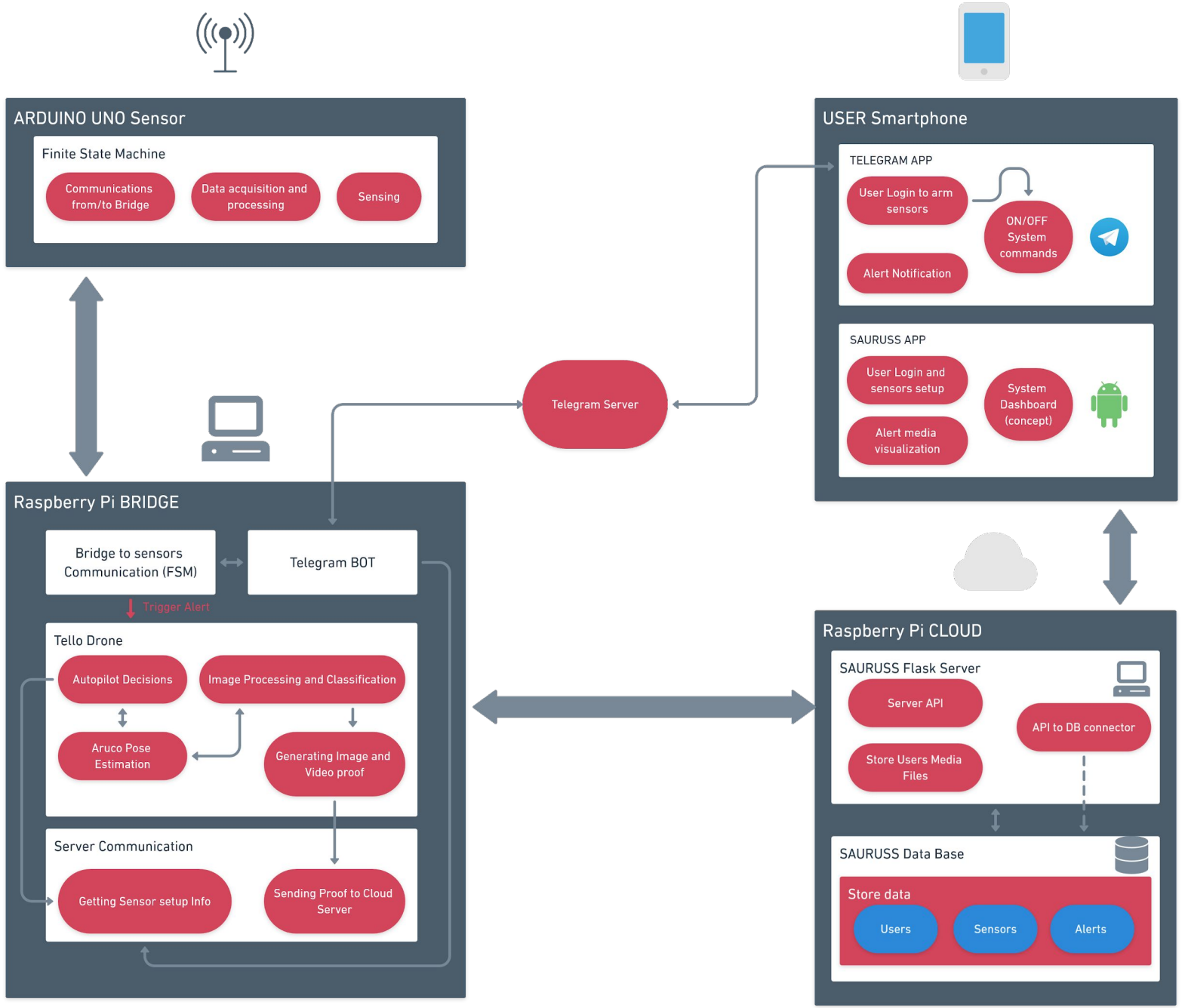
1. DIFFERENCES BETWEEN FINAL PRODUCT AND OUR PROTOTYPE
2. HARDWARE AND SOFTWARE USED FOR PROTOTYPE
3. FULL ARCHITECTURE WITH MAIN FUNCTIONALITIES

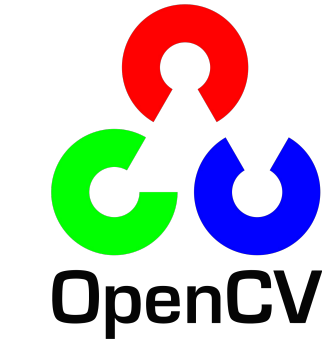
From Idea to Prototype: the differences

- We used Telegram Bot in conjunction with the companion app for demo purposes (it shows we can easily switch between using a third party service and our cloud server)
- We used a drone without a GPS and added Aruco Pose estimation to increase autonomous drive accuracy
- We used both Raspberry Pi4 and a Laptop pc for testing and debugging the bridge script

Simplified Architecture




ARDUINO UNO Sensor
Finite State Machine
Communications from/to Bridge
Data acquisition and processing
Sensing
USER Smartphone
TELEGRAM APP
User Login to arm sensors
ON/OFF System commands
Alert Notification
SAURUSS APP
User Login and sensors setup
System Dashboard (concept)
Alert media visualization
Telegram Server
Raspberry Pi BRIDGE
Bridge to sensors Communication (FSM)
Telegram BOT
Trigger Alert
Tello Drone
Autopilot Decisions
Image Processing and Classification
Aruco Pose Estimation
Generating Image and Video proof
Server Communication
Getting Sensor setup Info
Sending Proof to Cloud Server
Raspberry Pi CLOUD
SAURUSS Flask Server
Server API
API to DB connector
Store Users Media Files
SAURUSS Data Base
Store data
Users
Sensors
Alerts



HARDWARE AND SOFTWARE USED

Prototype

Hardware used	
Microcontroller (2x) (Sensors Modules)	Arduino UNO Rev. 3
Sensor Hub (Bridge)	Raspberry Pi 4
Server Cloud	Raspberry Pi 2B
Drone (UAV)	DJI Tello EDU
Smartphone (Client)	Android Smartphone
Microntoller Sensors, Comm. Devices, Actuators	PIR proximity sensor, Vibration Sensor,

Technical Talk 

1. SENSORS AND BRIDGE FSM FOR COMMUNICATIONS
2. CLIENT APP AND TELEGRAM BOT
3. BACKEND AND BRIDGE DRONE AUTOMATION

Sensors and Bridge FSM

Why use Finite State Machine patterns?

- Robust software
- Easy to change/add functionalities
- Applicable everywhere

Sensors main board: Arduino UNO Rev. 3

- Good for quick prototypes
- Cheap
- Reduced form factor
- Plenty of documentation
- Low power
- VS Code IDE and Arduino IDE

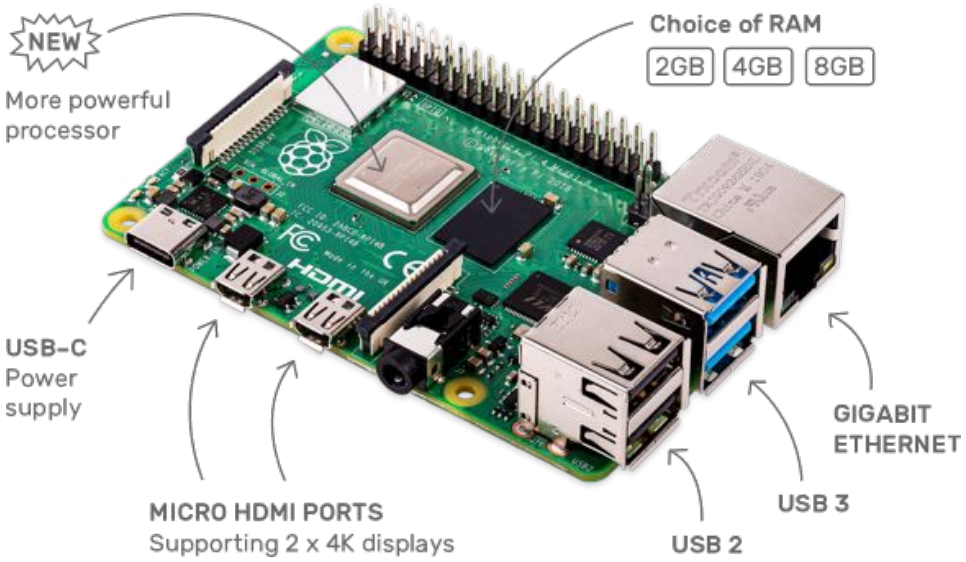
What does it need to do?

- Take data from sensors
- Elaborate data and decide if to send it
- Communicate reliably to Bridge/Listen for Bridge commands

SPECS

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Bridge: Raspberry Pi 4 Model B (4 GB RAM Model)



- Low cost
- Low power
- Powerful but small
- Can handle a little bit of image processing
- Wifi + Ethernet + Bluetooth all in one

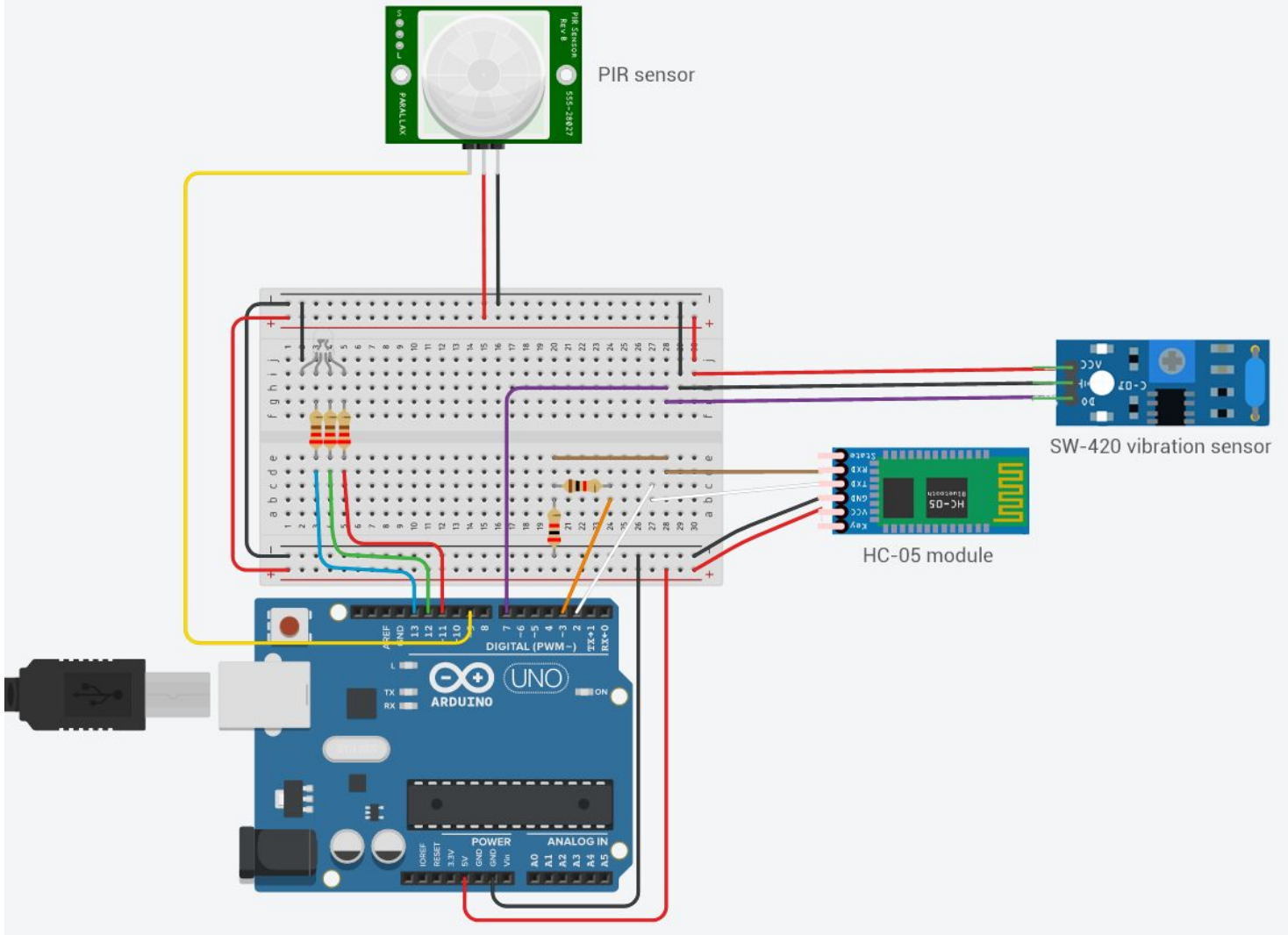
SPECS

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 4GB LPDDR4-3200 SDRAM
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header

What does it need to do?

- Communicate with Drone (wifi), Sensors (Bluetooth), Server Cloud and Telegram Server (ethernet)
- Handle actions when Sensor alert (activate Drone) or telegram bot message
- Handle Drone commands, image processing, pose estimation and CNN image classification

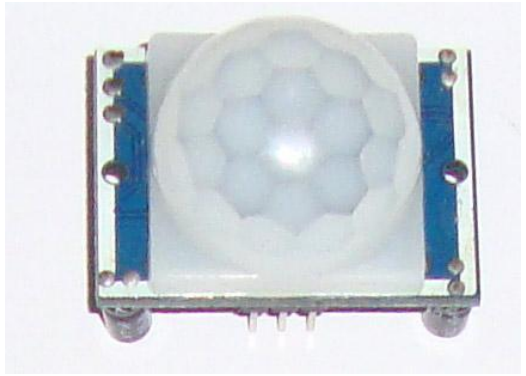
Sensor Prototype Circuit Schematic

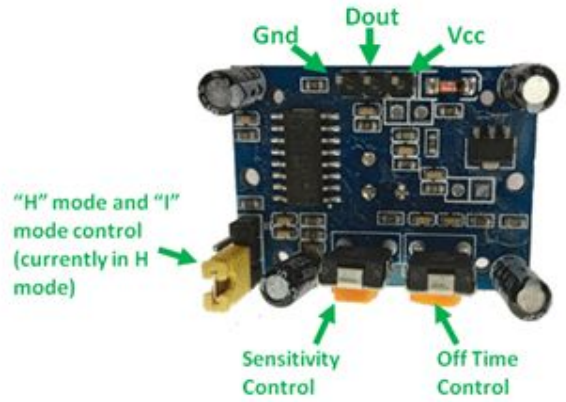


Hardware Used

Component Name	Quantity
Arduino UNO Rev. 3	1
Breadboard	1
Resistors	3 x (220 Ω) 1 x (1kΩ) 1 x (2kΩ)
RGB LED	1
HC-05 Bluetooth Module	1
PIR Proximity Sensor	1
SW-420 Vibration Sensor	1

Sensors: PIR





Pin Name	Description
Vcc	Input voltage is +5V for typical applications. Can range from 4.5V- 12V
High/Low Ouput (Dout)	Digital pulse high (3.3V) when triggered (motion detected) digital low(0V) when idle(no motion detected
Ground	Connected to ground of circuit

Repeatable(H) mode

In Repeatable(H) mode the output pin Dout will go high (3.3V) when a person is detected within range and goes low after a particular time (time is set by "Off time control" potentiometer)

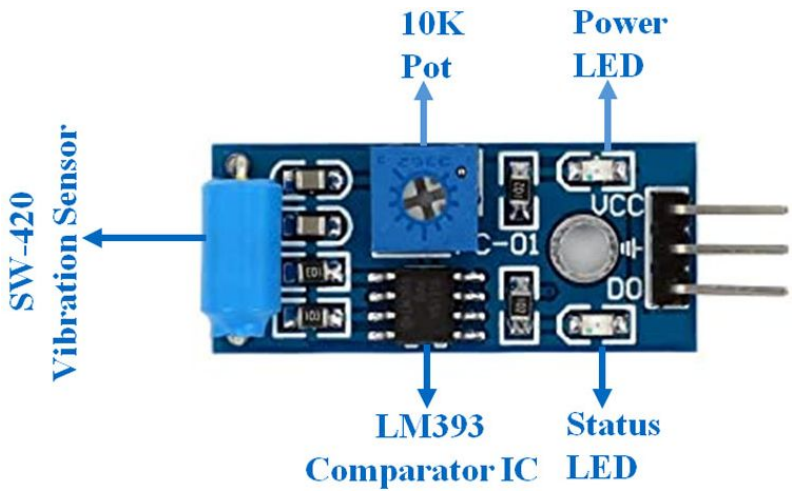
Non- Repeatable(L) mode

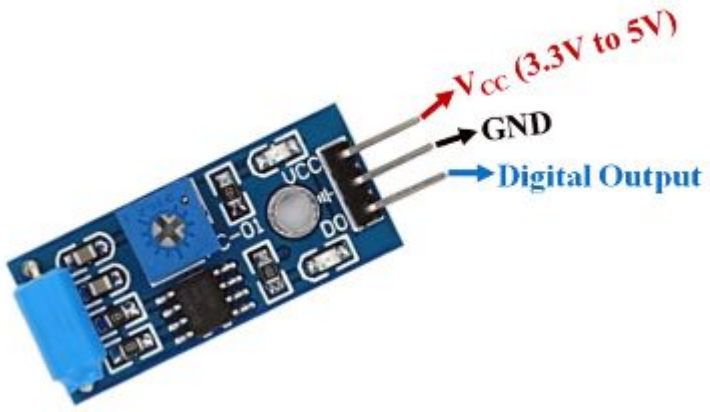
In "I" mode the output pin Dout will go high (3.3V) when a person is detected within range and will stay high as long as he/she stays within the limit of the Sensors range. Once the person has left the area the pin will go low after the particular time which can be set using the potentiometer

- Cover distance of about 120° and 7 meters
- Low power consumption of 65mA
- Output voltage is High/Low (3.3V TTL)
- Can understand if there is a living being or not*

*All object with a temperature greater than the absolute zero emit light radiation

Sensors: Vibration

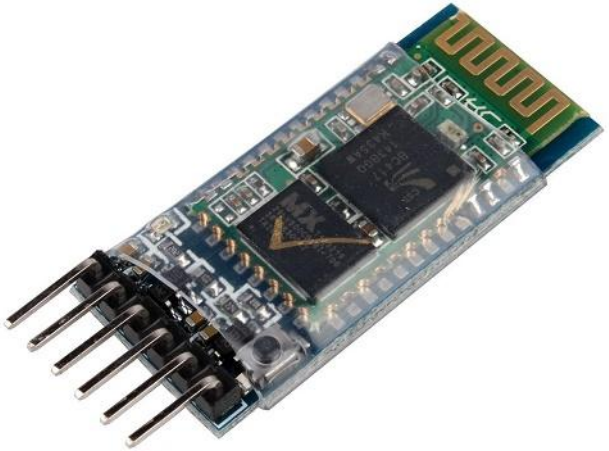


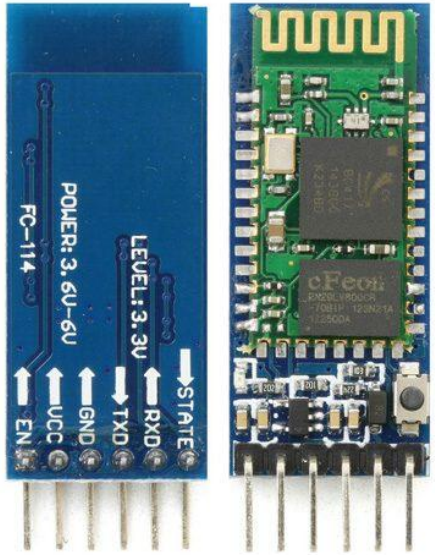


- Output at DO is HIGH when vibration is sensed (status led blinks)
- Output is LOW otherwise
- Cheap and easy to install thanks to pre-built bolt hole
- Low current consumption (15mA)
- 10 K Pot var. resistor used to set sensibility

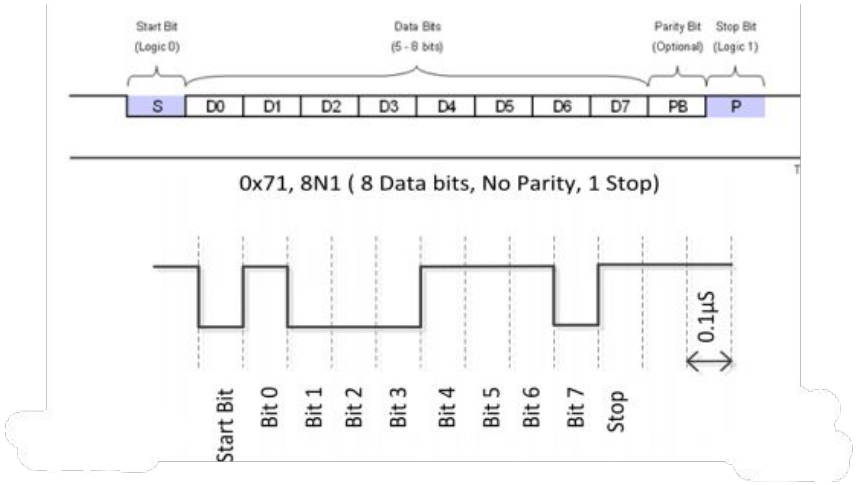
Pin Name	Description
VCC	The Vcc pin powers the module, typically with +5V
GND	Power Supply Ground
DO	Digital Out Pin for Digital Output.

Communication: Bluetooth HC-05 Module





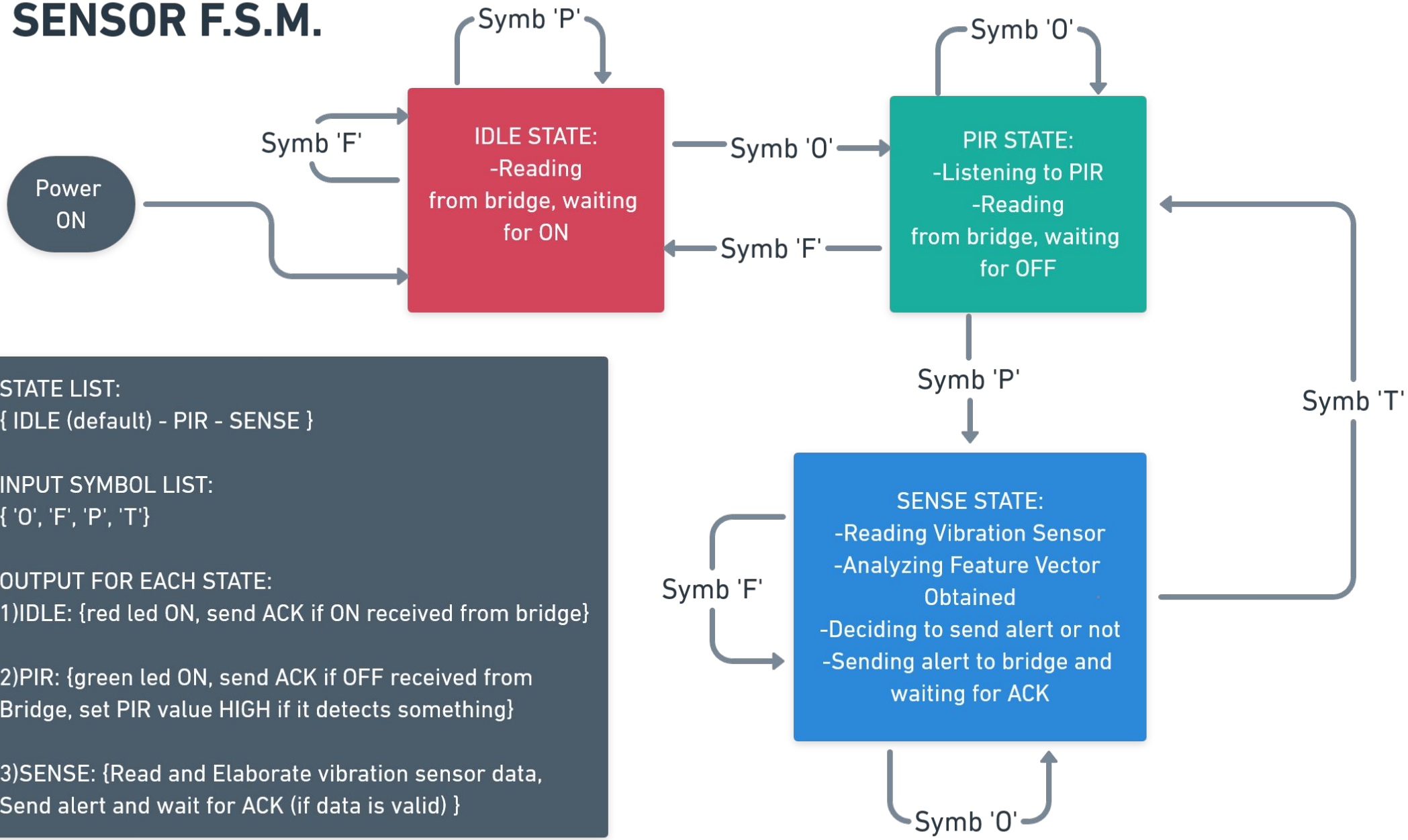
- UART Serial to BT 2.0 communication
- Based on CSR Bluecore 04 External chip
- 3,6V to 6V power supply tension, 3.3V data lines (TX-RX), 50mA current consumption
- Cheap and easy to use with Software Serial library for Arduino



Asynchronous serial transmission (UART)

Clock is synchronized with data encoding but sender and receiver needs to have same BAUD rate

SENSOR F.S.M.



STATE LIST:
{ IDLE (default) - PIR - SENSE }

INPUT SYMBOL LIST:
{ '0', 'F', 'P', 'T'}

OUTPUT FOR EACH STATE:
1)IDLE: {red led ON, send ACK if ON received from bridge}

2)PIR: {green led ON, send ACK if OFF received from Bridge, set PIR value HIGH if it detects something}

3)SENSE: {Read and Elaborate vibration sensor data, Send alert and wait for ACK (if data is valid) }

SENSOR FSM Pseudo - Code

Imports, global costant & variables declaration

void setup

void loop {

Read PIR

Read for incoming messages every T seconds

UpdateState( )

present_state = future_state

UpdateOutput( )

}

USED FUNCTIONS

Data Analysis Functions

ReadSensors( )

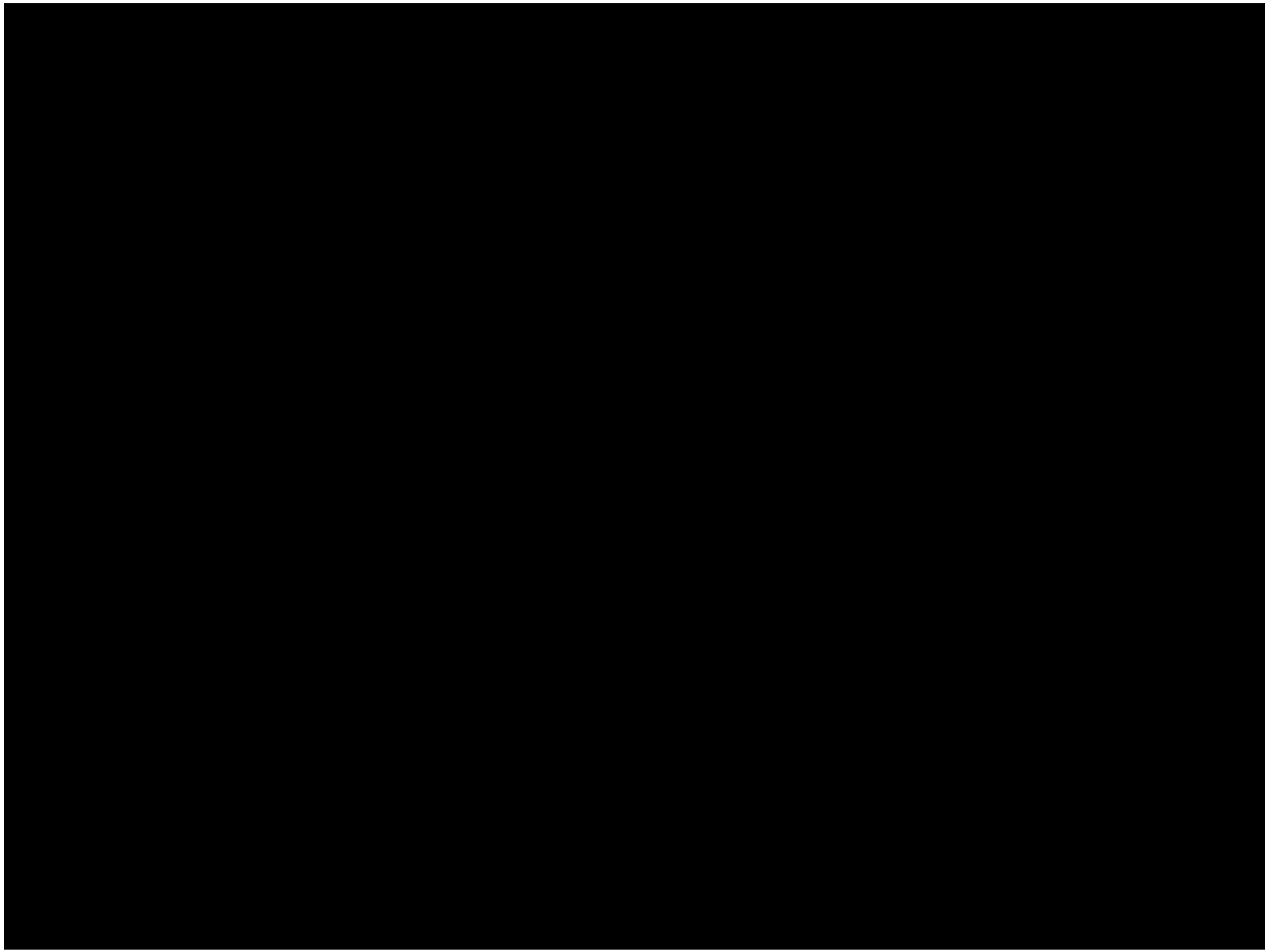
Analyze_Data( )

Communications Functions

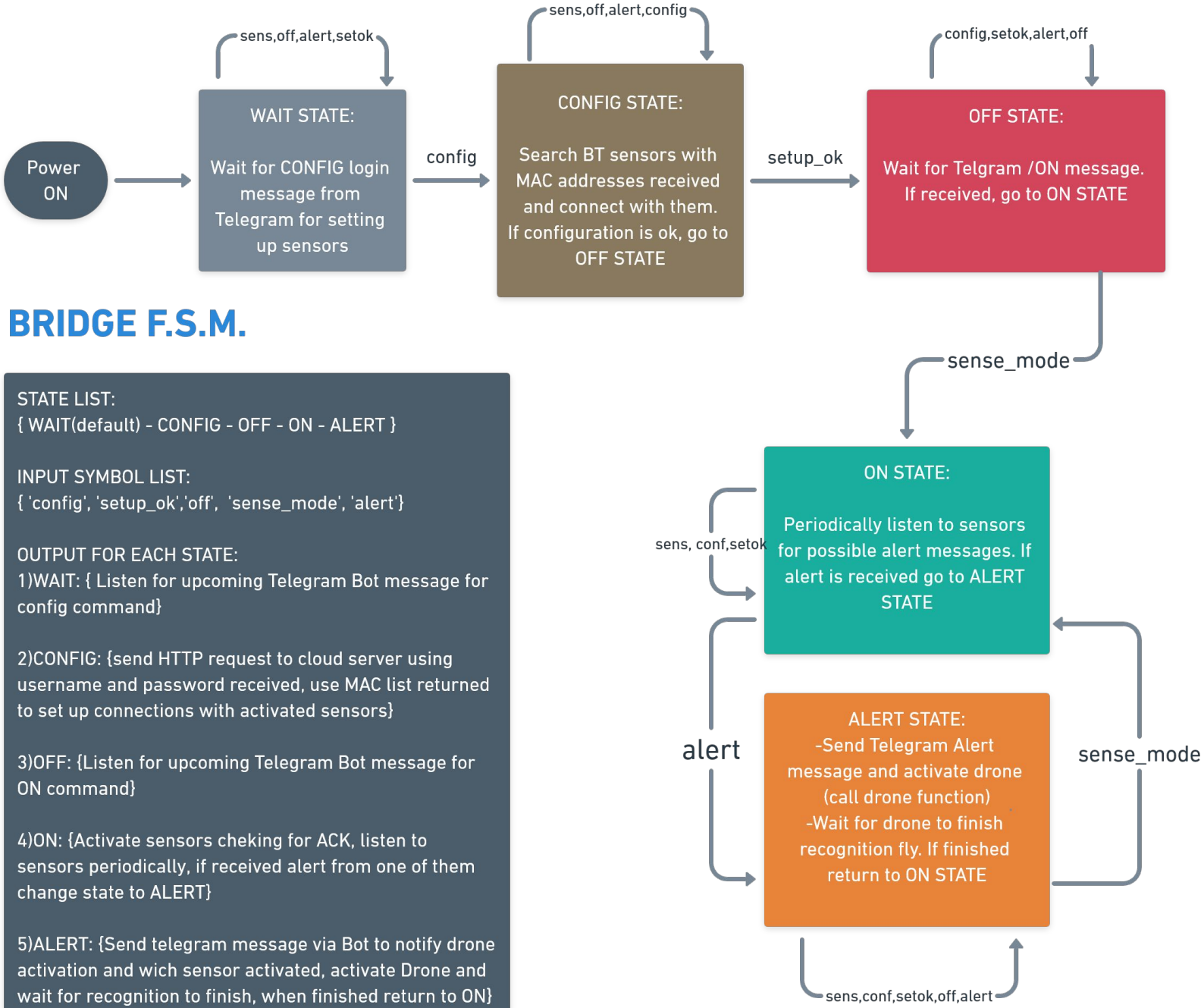
Read_Data( )

Send_ACK( )

Send_and_wait( )



DEMO VIDEO



BRIDGE F.S.M.

STATE LIST:
{ WAIT(default) - CONFIG - OFF - ON - ALERT }

INPUT SYMBOL LIST:
{ 'config', 'setup_ok','off', 'sense_mode', 'alert'}

OUTPUT FOR EACH STATE:
1)WAIT: { Listen for upcoming Telegram Bot message for config command}

2)CONFIG: {send HTTP request to cloud server using username and password received, use MAC list returned to set up connections with activated sensors}

3)OFF: {Listen for upcoming Telegram Bot message for ON command}

4)ON: {Activate sensors cheking for ACK, listen to sensors periodically, if received alert from one of them change state to ALERT}

5)ALERT: {Send telegram message via Bot to notify drone activation and wich sensor activated, activate Drone and wait for recognition to finish, when finished return to ON}

BRIDGE FSM Pseudo - Code

Imports, global costant & variables declaration

methods definitions

main {

Instantiate Bridge

Start Thread from Bot.py script

While(True):

UpdateState( )

On_Enter_Actions( )

present_state = future_state

UpdateOutput( )

}

USED FUNCTIONS/METHODS

Bot.py meyhods*

Bridge_Config( )

Setup_BT_Connection( )

Setup_Serials( )

Bridge methods/comm.

Set_Sensors_ON( )

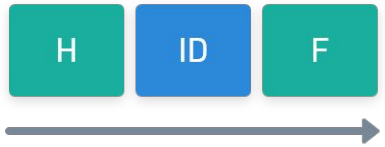
Set_Sensors_OFF( )

Listen_for_Alert( )

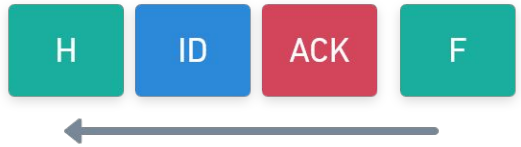
*More on that later :)

PACKET PROTOCOL USED BETWEEN SENSORS AND BRIDGE

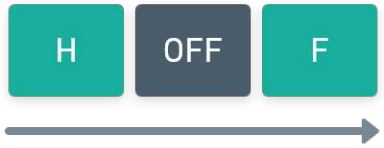
ON/SETUP FROM BRIDGE



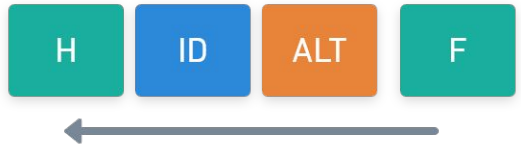
ACK FROM SENSOR



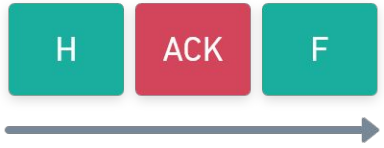
OFF FROM BRIDGE

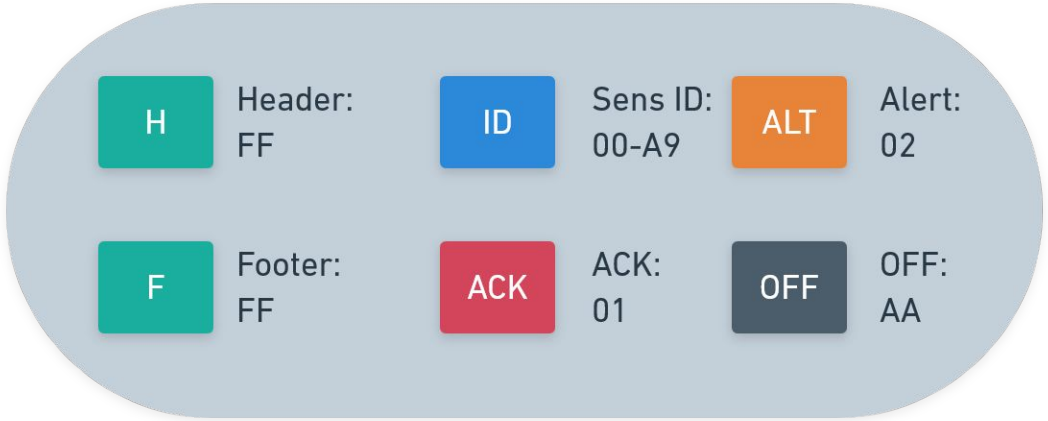


ALERT FROM SENSOR



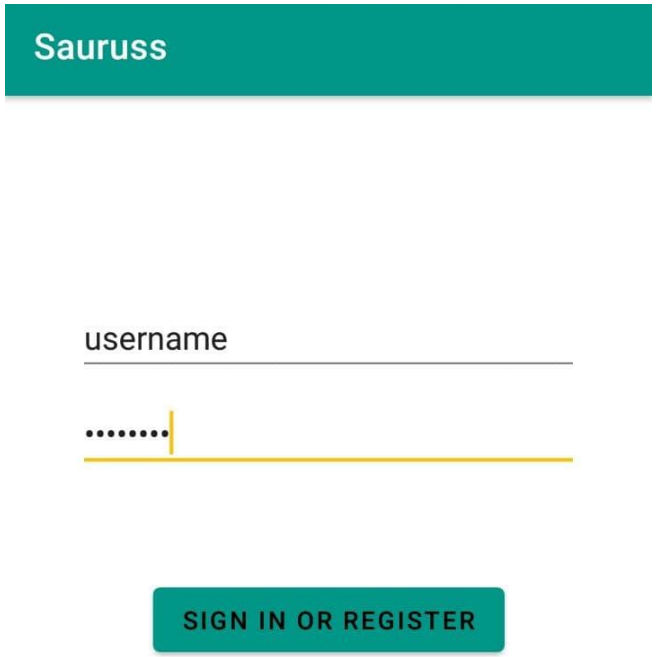
ACK AFTER ALERT FROM BRIDGE

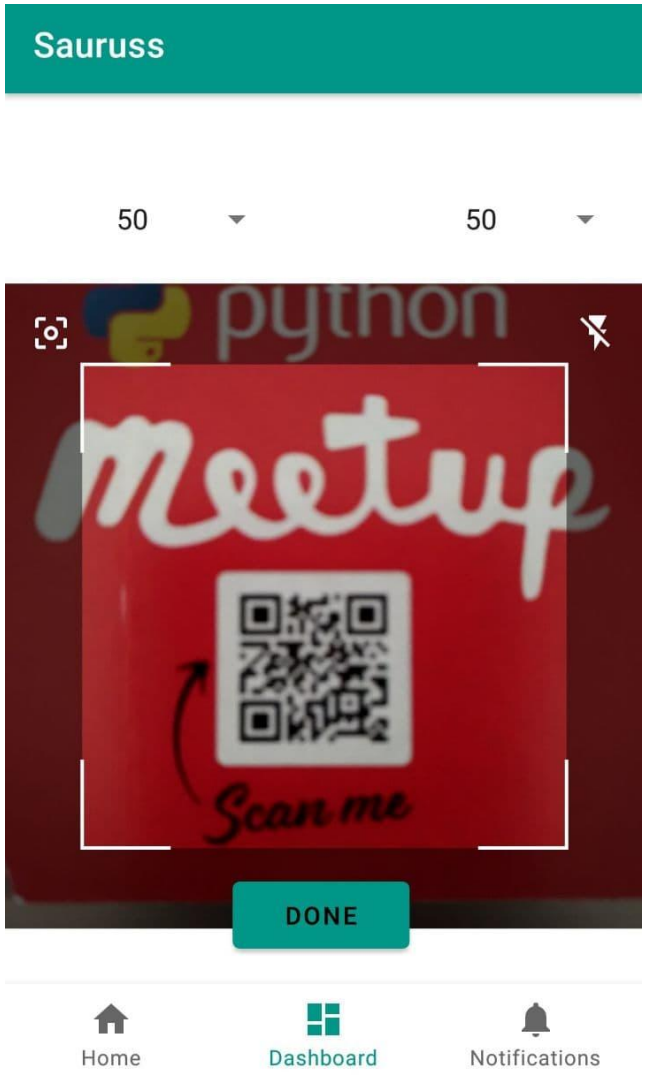


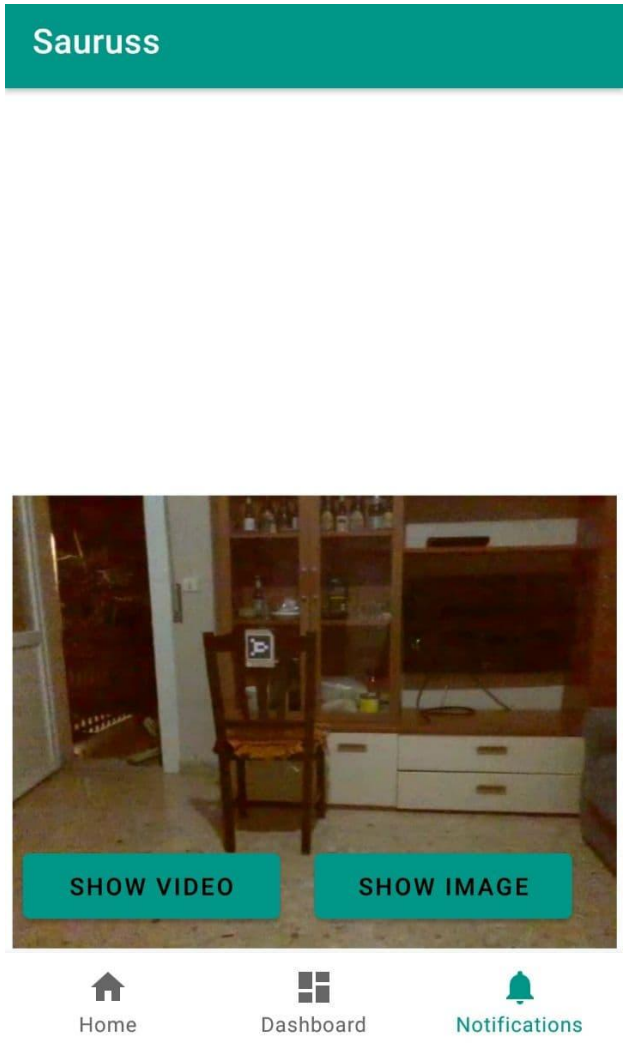


Client APP and Telegram Bot

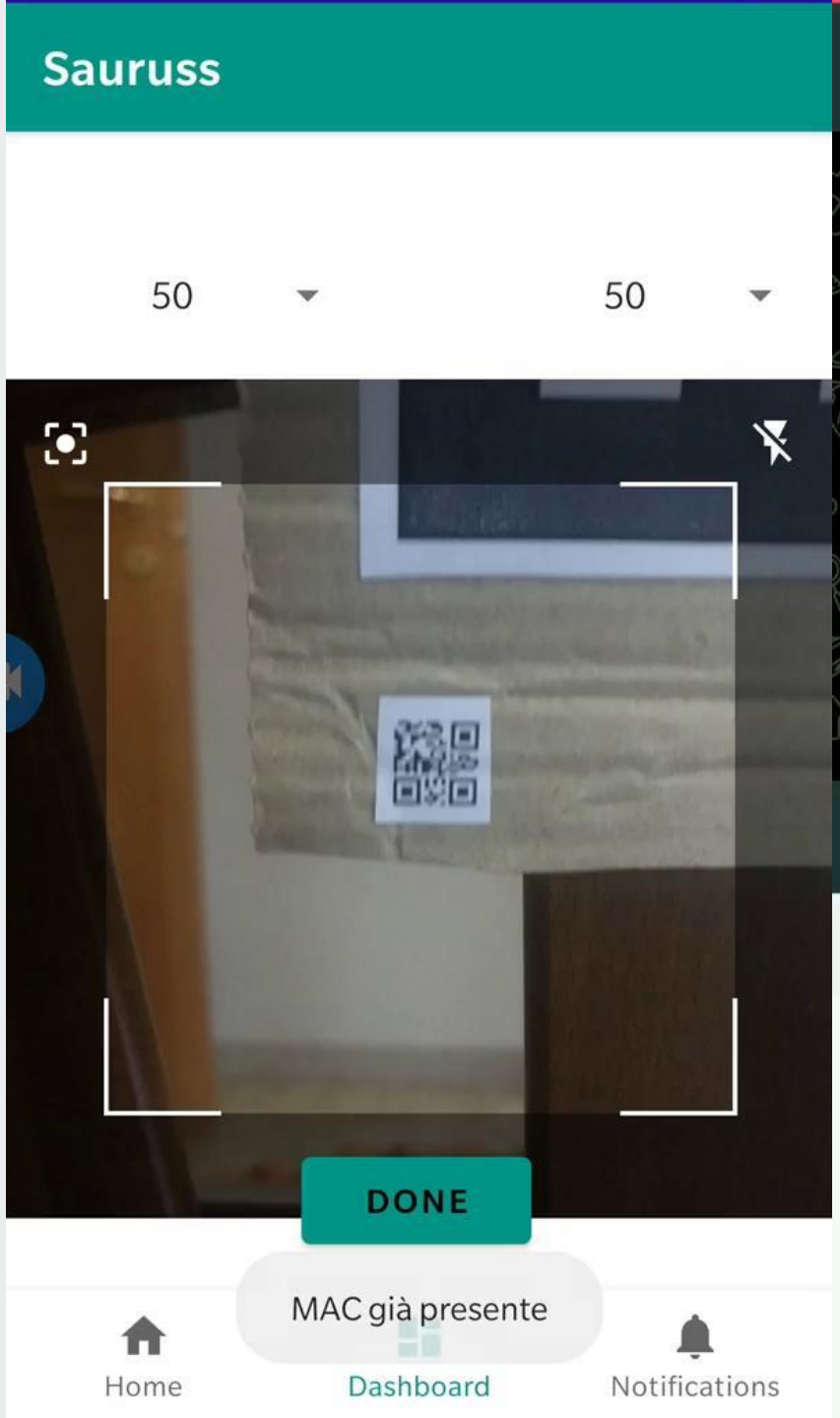
Smartphone Application



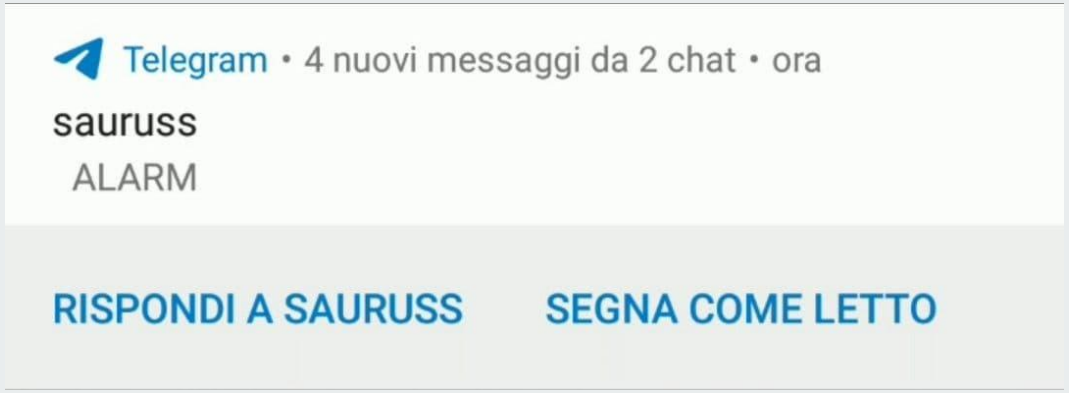


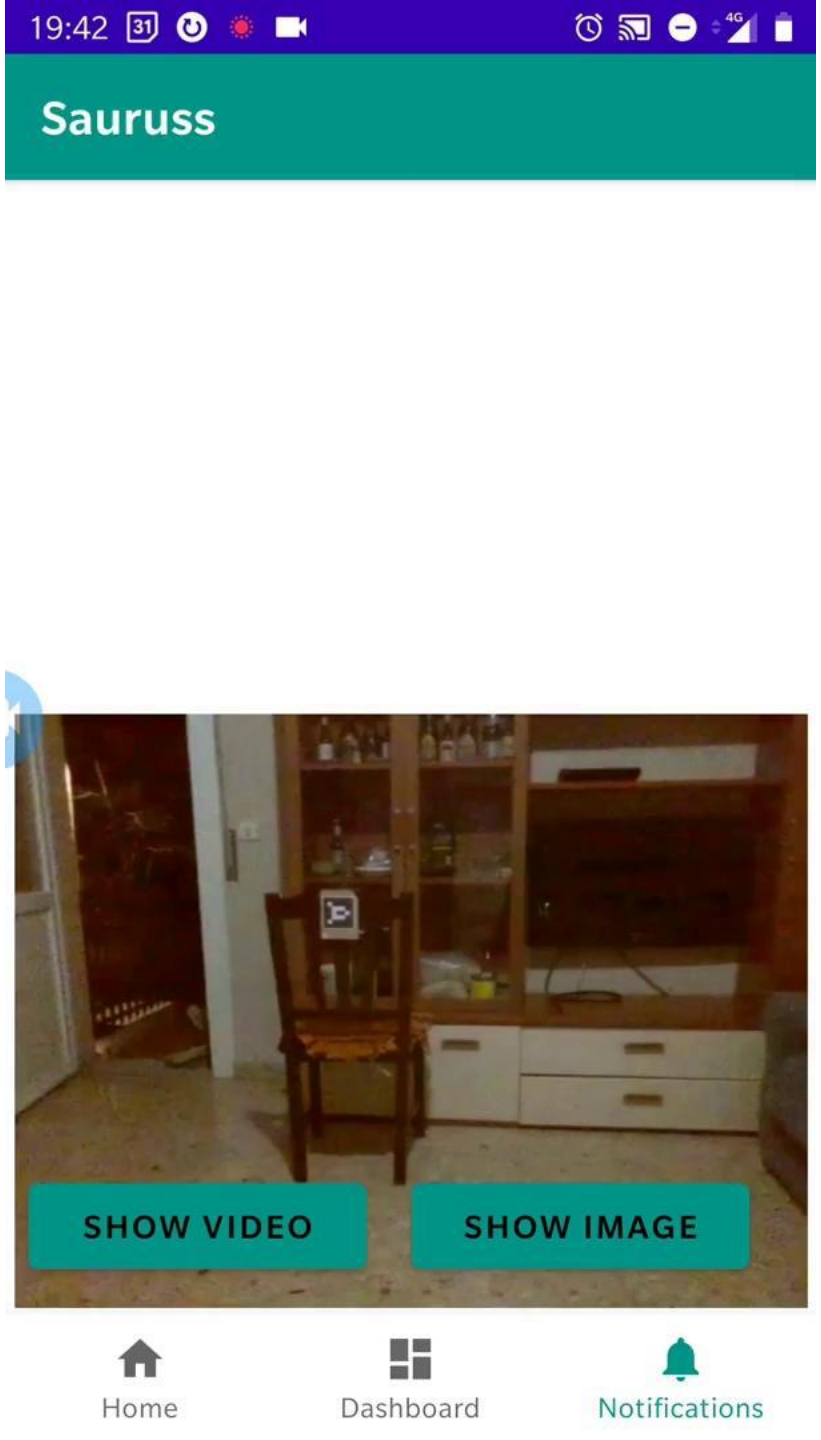


Setup & Configuration



Notification





API – Application side (general)

Name	function
AndroidManifest	required to declare the app's package name, *components* of the app and necessary *permissions* for the app to work properly
UserInfo	used to *set* and *retrieve* the global variables of the current session, that needs to be shared throughout all the app
LoginActivity*	start activity of the app. Handles the login of the user and saves cookies.
MainActivity	redirects to home page of the app and handles the whole bottom navigation
Home	handles home view to show stats and/or recap of current state (once enough data will be collected)
Dashboard*	setup/modify *position* and *id* of the sensor used in the project
Notification*	shows last video/image captured by the drone

*next slide for specific info

API – Application side (specific)

File	Function	Request
LoginActivity	onCreate( )	HTTP POST request to submit **login** form or if response != 200 HTTP POST to **register** a new user
Dashboard Fragment	scanFunction( )	on QR code decoded makes **HTTP POST** request to submit the decoded value(MAC address of the device) to **sensors**
	dbCall(parameters)	*parameters*: MAC address and relative positions of the sensor manually selectionable. *does*: on clicking the button "done" makes an **HTTP POST** request to submit the parameters to **distances**
Notification Fragment	getFileName( )	**HTTP GET** requests to get the **timestamp** and **ID** of the sensor that triggered an alarm and so the **filename** of video and image captured by the drone
	downloadFile(*Url,saveDir*)	**HTTP GET** request to download **video** and **image** captured by the drone on alarm trigger

BOT Telegram API

Function	Description
ON_Command	on the user command "**/on**" puts FSM state to *sense_mode*, therefore turning on the sensors
OFF_Command	on the user command "**/off**" puts FSM state to *off*, therefore turning off the sensors and shutting down the alarm. (User should send this if has already checked the view notifications on the app and it's all ok)
send_alert_message*	when an *alert* occurs, makes a request to the telegram server in order to **notify the user**
send_bot_message*	send feedbacks to the user(through a request to telegram server) about sensors having been switched on succesfully or not
get_new_message	on configuration, user has to type *"config"* + " " + *username* + " " + *password* on the bot chat. this function checks if all is typed correctly and then uses username and password to retrieve the mac adresses from the db and configure their connection with the bridge

*each request to the telegram server requires chatID and BOTKEY

Backend and Bridge Drone Automation

Raspberry Pi bridge

Bridge is the *center of the drone's decisions* and also of the frame elaboration done by the camera. From here, all the HTTP requests to the SAURUSS flask server are sent.

This happens when an *alarm event* is generated somewhere in our sensor network.

Visual and Object recognition activities are based on *AI algorithms*.

API – Bridge [PACKAGE] HTTP_Session:

SESSION_SERVICE	REQUEST	DESCRIPTION
doLogin(user,password)	s.post(URI + “login/<user>/<psw>/”)	For each request, a login by the bridge of a specific user is required
getDistances()	s.get(URI+’distances/’)	A request to update the new sensor coordinates
uploadMediaFile (sensore,data,file)	s.post(URI+ ‘upload/<sensor>/<data>/’, file={‘media’:<file>})	Sending photos and videos in the moment of intrusion by the drone
sendAlarmBySensorID (sensor)	s.post(URI+ ‘alarms/<sensor>/”)	Sending the alarm with timestamp
getTimestamp(sensor)	s.get(URI+’alarms/timestamp/ <sensor>/’)	Returns the timestamps of the last alarm
URI	“http://thomasgar.ns0.it/”	Server SAURUSS

API – HTTP protocol

Type of Request	Route	Description
POST	/login/{username}/{password}	-
GET	/logout	-
POST	/registration/{nomeutente}/{password}/{phone}/	SQLAlchemy provides encryption
POST	/upload/{alarm_id}/{date}/	Store media files on SAURUSS server
GET	/download/{date}/{filename}/	Get the URL on DB and then start to download from server
GET	/alarms/	It provides the history of the past alarms
POST	/sensors/{sensor_id}/{latitude}/{longitude}/	Insert a new sensor on DB
POST	/distances/{sensor_id}/{x_position}/{y_position}/	Changes a sensor distances from the drone home

E/R database MySQL

The modularity of implementation of the connection to the database allows it to be accessible anywhere and therefore can reside on the server or on another separate machine.

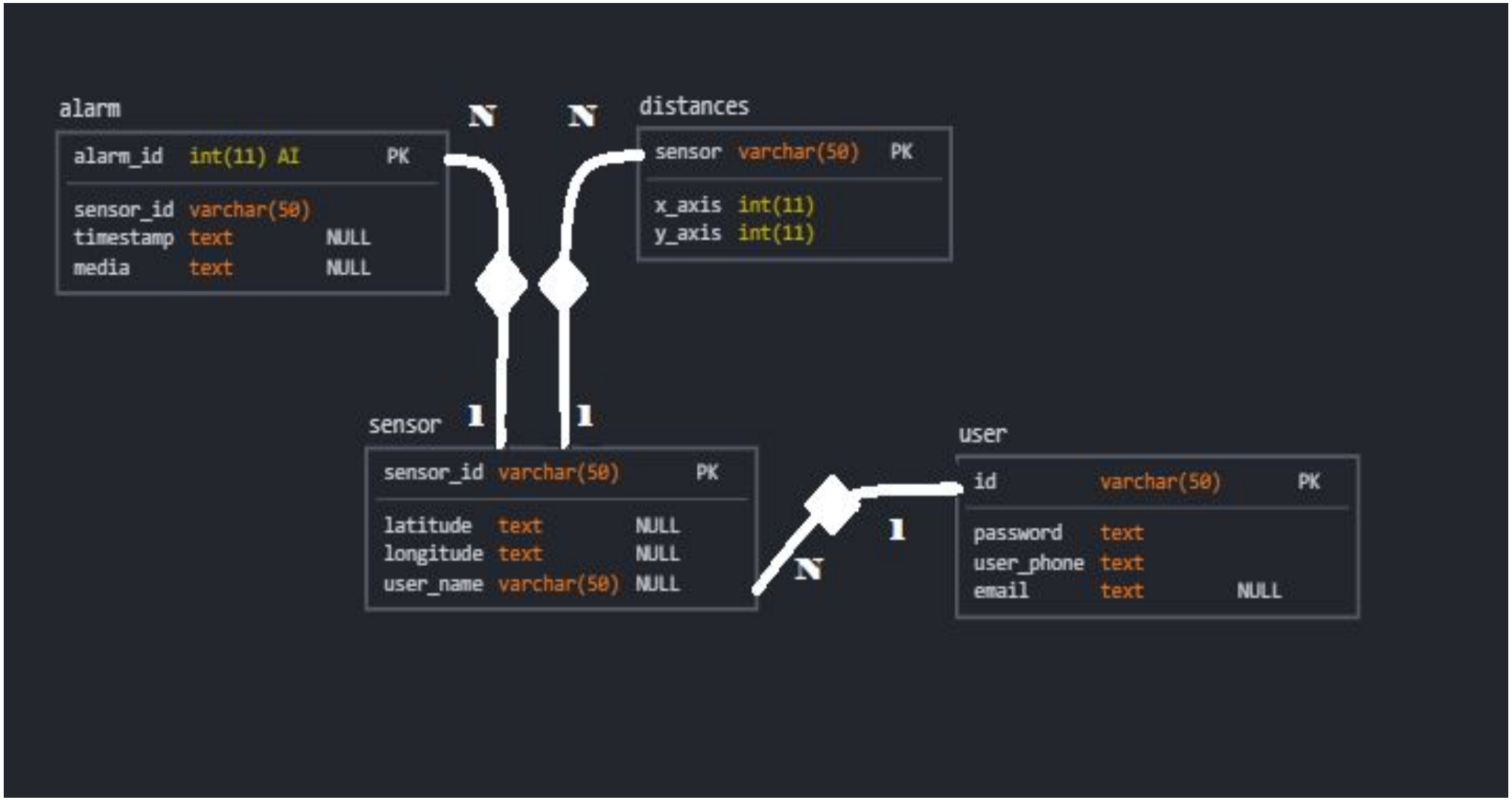
User (id, VARCHAR(50) PK; password, TEXT; user_phone, TEXT)

Sensor (sensor_id, VARCHAR(50) PK, user_id, VARCHAR(50) FK(User); latitude, TEXT; longitude, TEXT)

Distances (sensor, VARCHAR(50) FK(Sensor); x_axis, INT; y_axis, INT)

Alarm (alarm_id INTAUTOINCREMENT PK, sensor_id VARCHAR(50) FK(Sensor), timestamp, DATETIME, media, TEXT)

E/R database MySQL



Threads description

In order to perform the self-piloting movements of the drone, for the simultaneous execution of the visual recognition (objects and aruco pose estimation in 3D space), and finally for the handler of the Telegram bot it was necessary to implement a multithreaded code, as well as a solid task orchestration.

[Package] Autonomous Driving

-Recognition Thread: YOLO performs a neural network described later

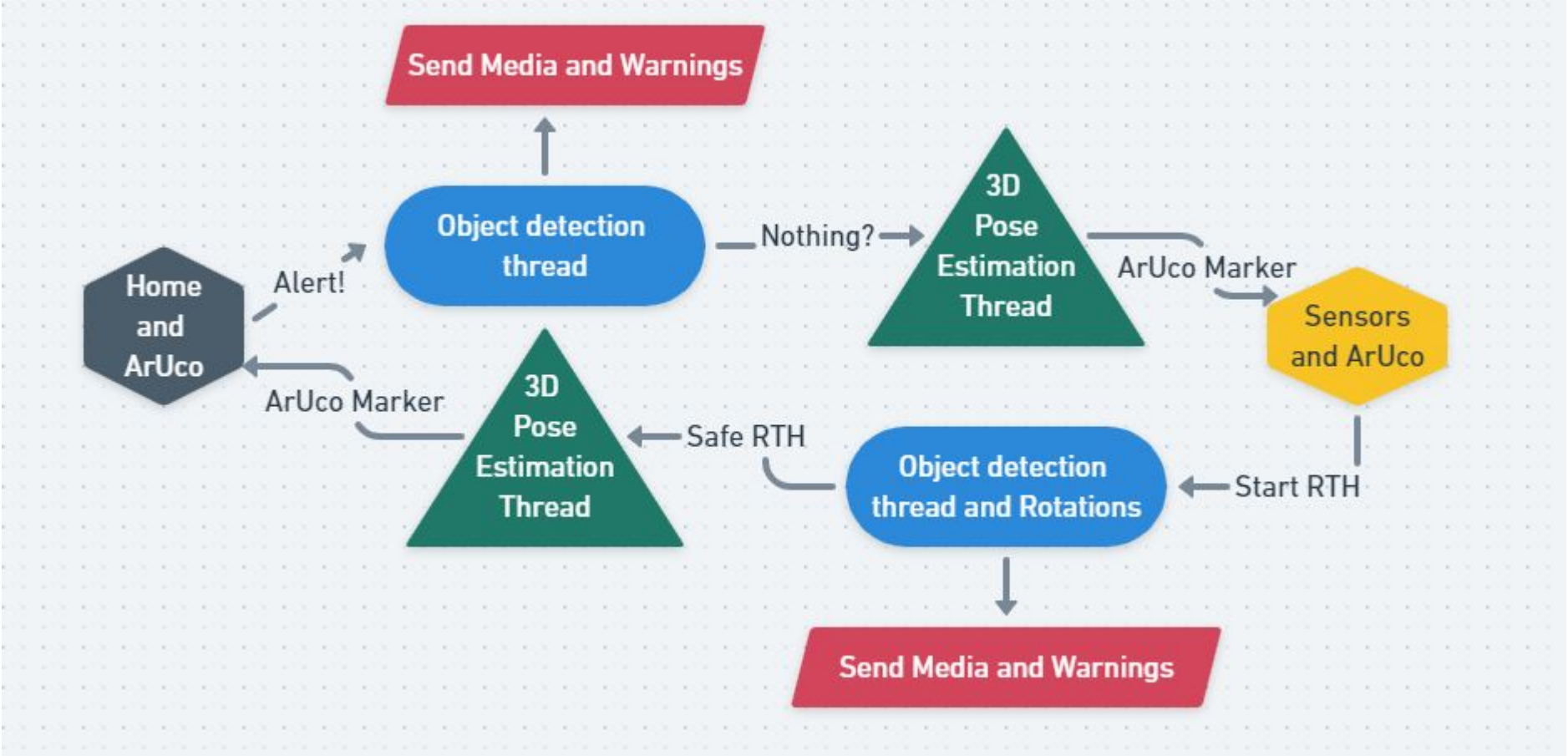
-Rotation Thread: These threads work together

- RTH(Recognition) : YOLO

[Package] Bridge Comm

- Bot (telegram): It constantly polls the Telegram server

Graphic description



YOLO: [PACKAGE] Object_Detection

The deep learning side of the project is definitely YOLO, a convolutional neural network that allows objects to be recognized thanks to computer vision algorithms.

In detail:

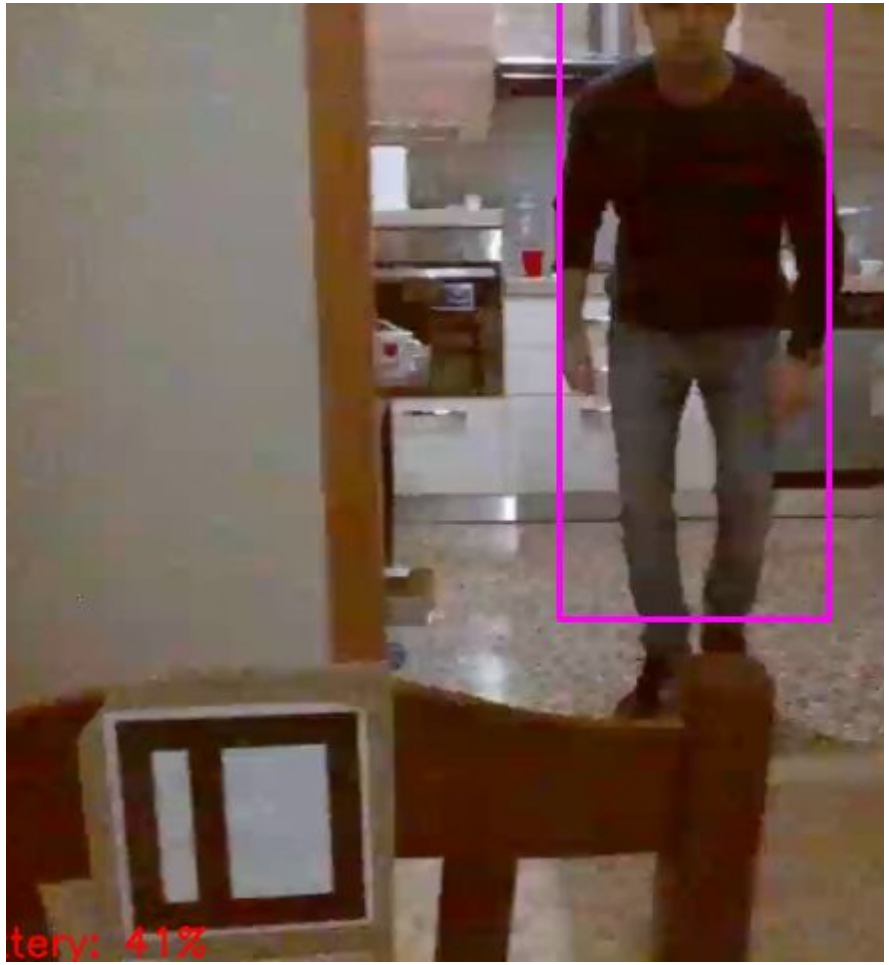
-Configurations: "yolov3-tiny.cfg"

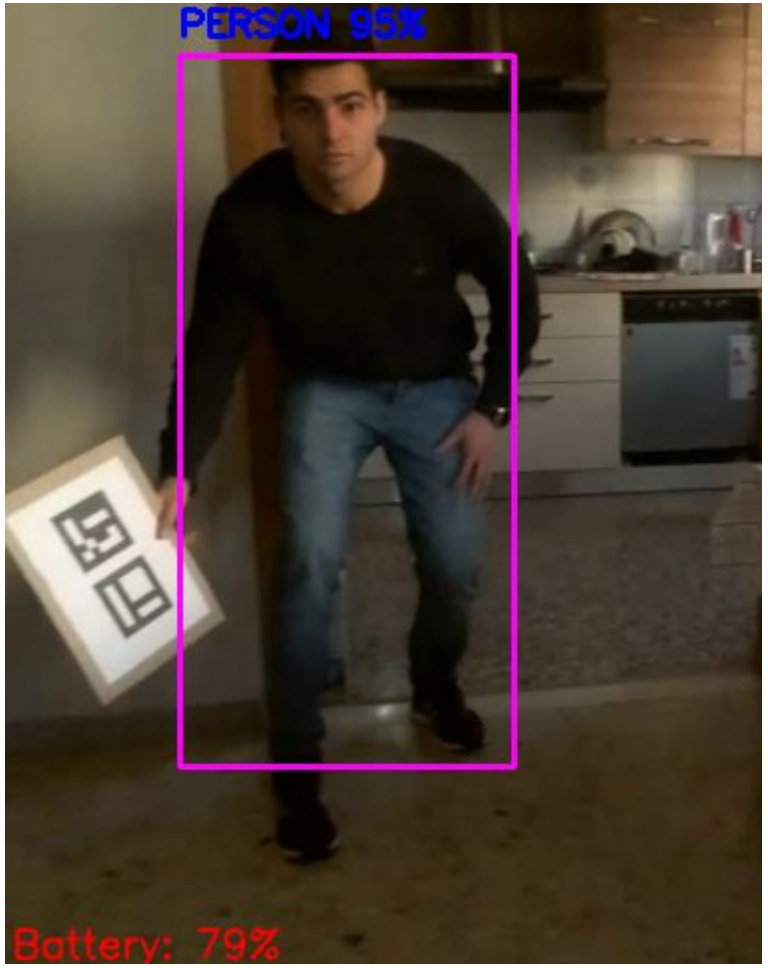
- Weights: "yolov3-tiny.weight"

It was decided to use the TINY version of the weights and the configuration in order to cope with the bridge calculation limits.

At the exact moment of recognition of a person, a photograph is taken as proof and a short video.

YOLO examples





3D Pose Estimation

To cope with the problem of missing GPS and the problems of instability of the drone it was necessary to implement a system of pose estimation in the operating space. Position estimation is of great importance in many computer vision applications: robotic navigation, augmented reality and many others.

Used technologies:

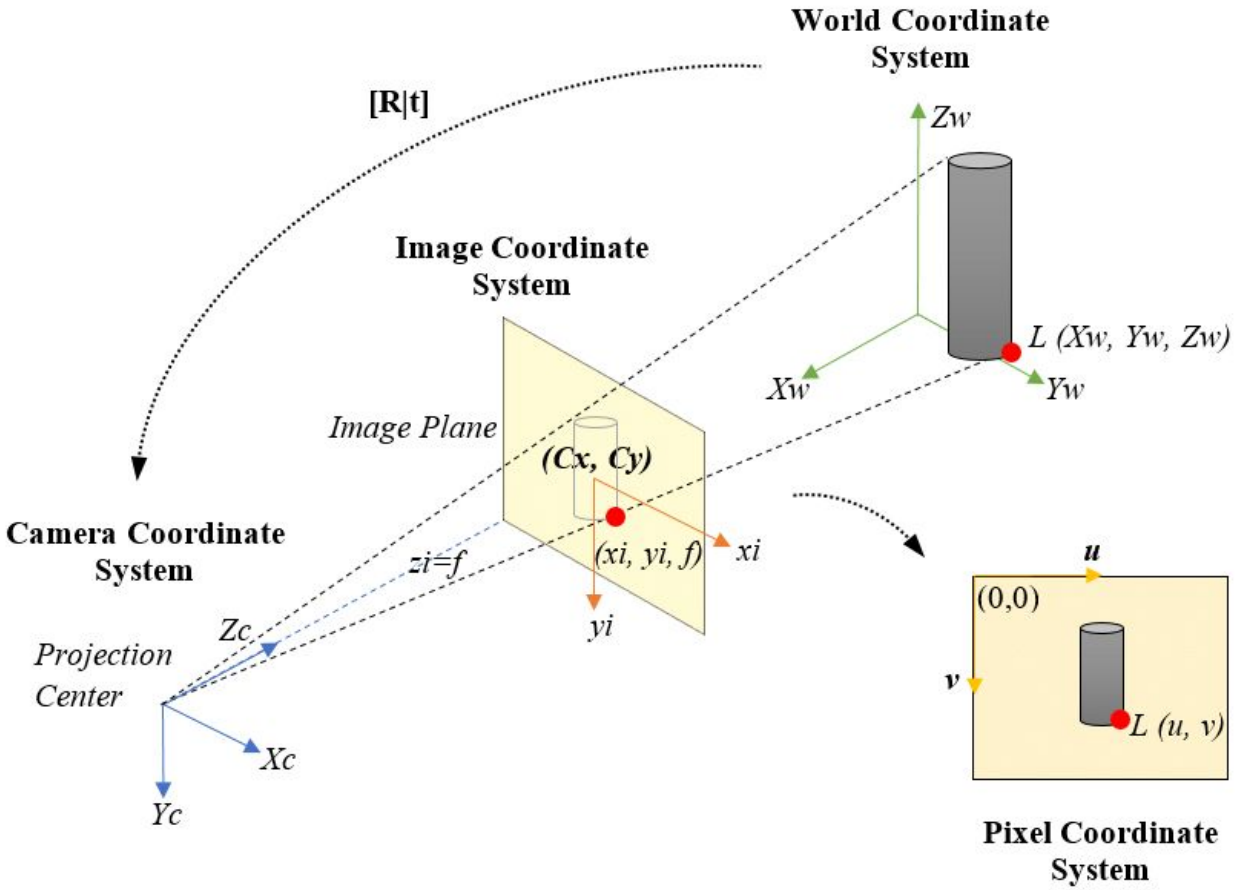
-Camera Calibration (openCV)

-ArUco markers (openCV)

-Euler Angles Rotations (openCV)

Pinhole camera model

The **pinhole camera model** describes the mathematical relationship between the coordinates of a point in three-dimensional space and its projection onto the image plane of an ideal pinhole camera, where the aperture of the camera is described as a point and no lenses are used to focus the light.



Distortion problem

The pinhole camera model does not include, for example, geometric distortions or blurring of blurred objects caused by lenses and apertures of finite size.

Camera Calibration

Calibration is necessary in order to derive the intrinsic parameters of the camera, such as:

- **Distortion of the image** (lens curvature).

$$Distortion\ coefficients = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

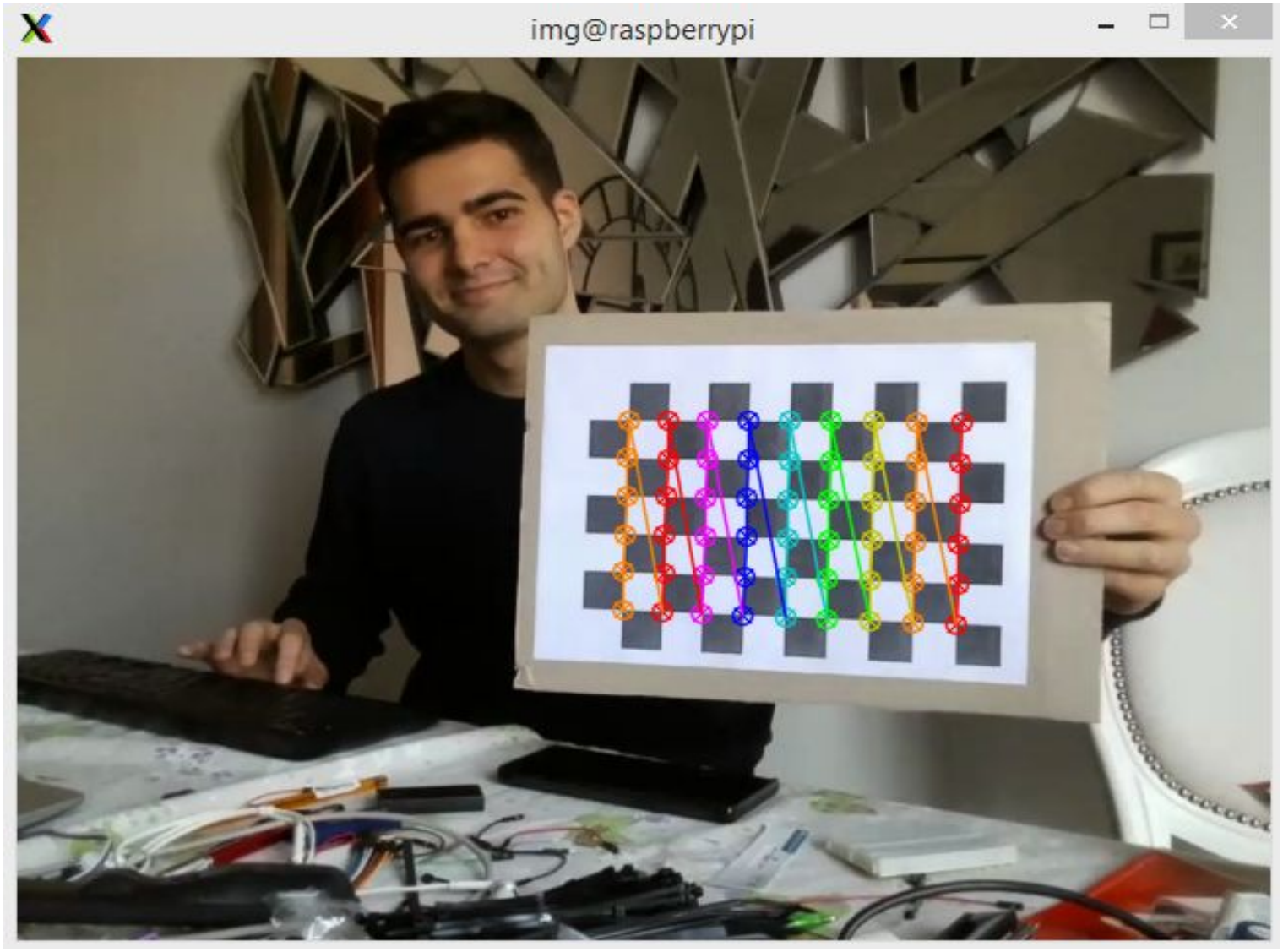
-**Focal length**

-Optical center

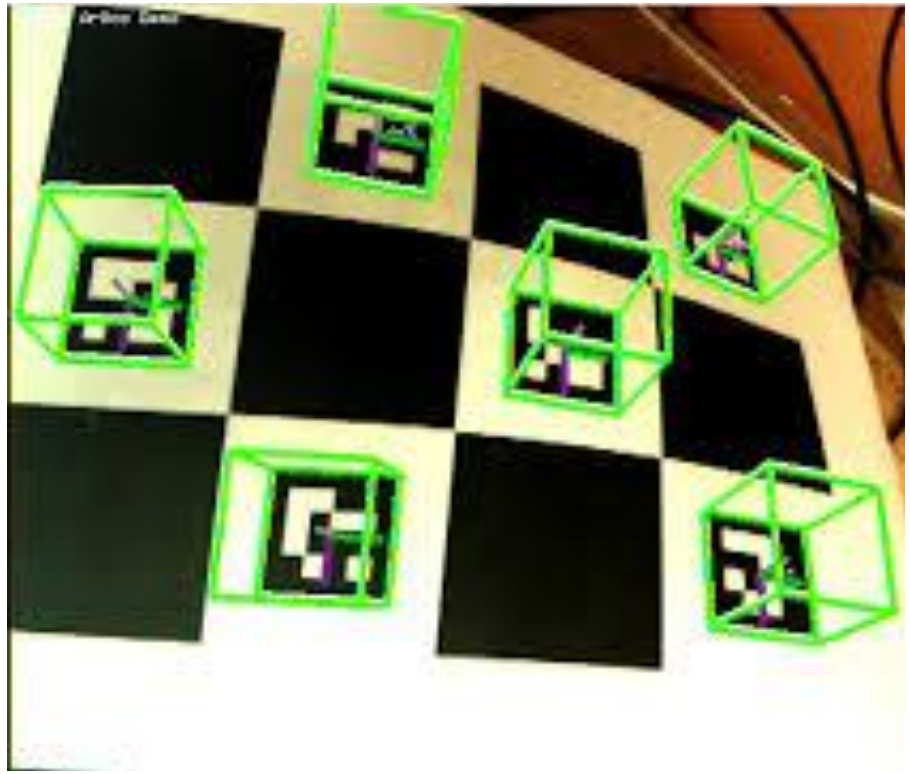
$$camera\ matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Chessboard

The calibration of the drone camera was done using a 9x6 chessboard (number of intersections between chess).



ArUco Markers



The main advantage of these markers is that a single marker provides enough matches (its four corners) to achieve the camera pose. In addition, the internal binary coding makes them particularly robust, allowing the possibility of applying error detection and correction techniques.

ArUco marker pose estimation

Pseudo code

rot = aruco.estimatePoseSingleMarkers (corners, marker size, camera matrix, camera distortion)

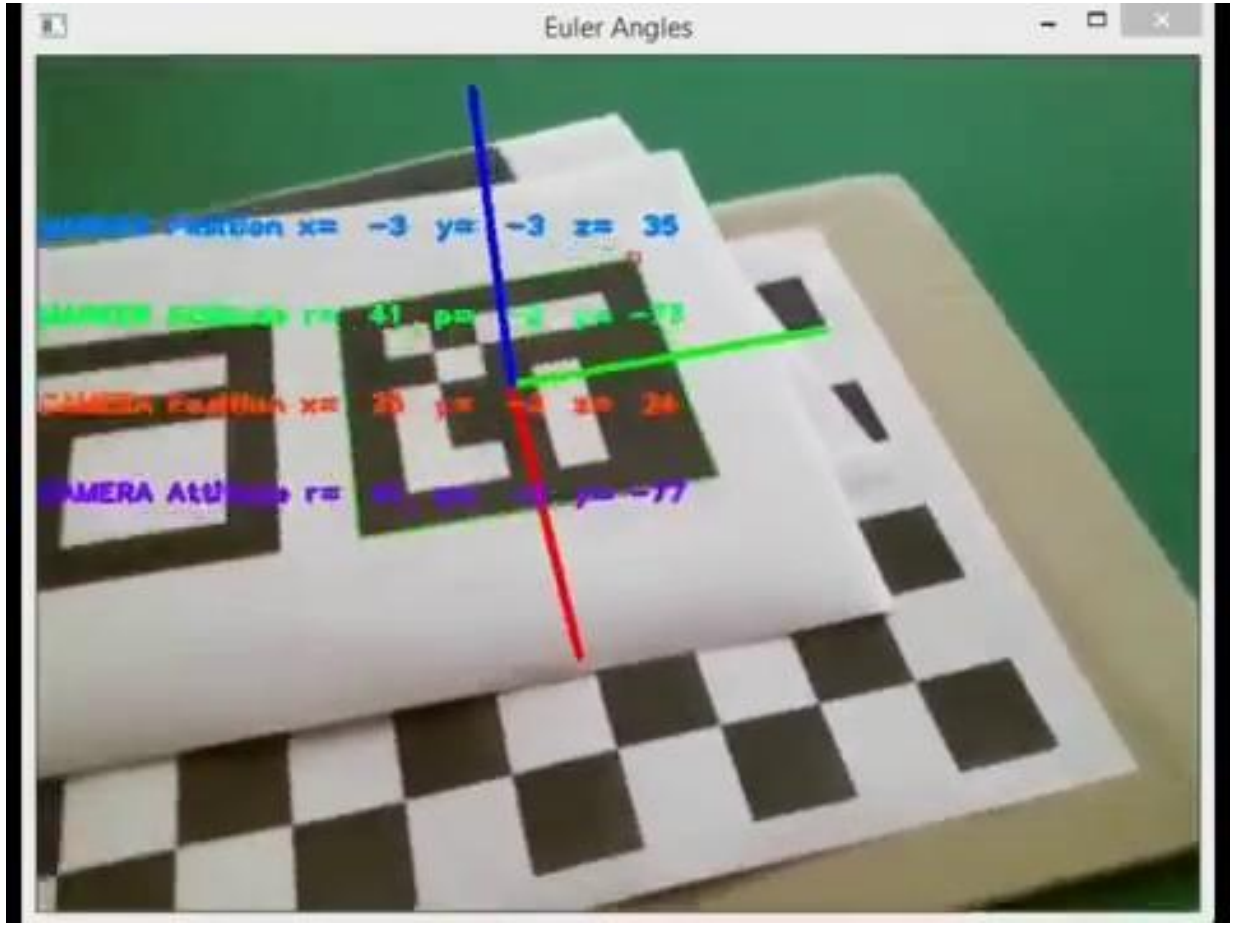
returns a matrix from which the rotation and translation vector can be extrapolated (rvec, tvec)

Rot_matrix = cv2.Rodriguez(rvec)

defines rotation on an axis with a vector (v * theta)

roll, pitch, yaw = rotationMatrixToEulerAngles(Rot_matrix.T)

ArUco marker pose estimation



DJI Tello Drone

Tello is the drone was used in the prototype. It's a small drone (80g) and it's easy to program.

Specs:

- **Weight: 87g (Propellers and Battery Included)**
- **Dimensions: 98×92.5×41 mm**
- **Propeller: 3 inches**
- **Built-in Functions: Range Finder, Barometer, LED, Vision System, 2.4 GHz 802.11n Wi-Fi, 720p Live View**
- **Port: Micro USB Charging Port**

- **Photo: 5MP (2592x1936)**
- **FOV: 82.6°**
- **Video: HD720P30**
- **Format: JPG(Photo); MP4(Video)**
- **EIS: Yes**

Tello SDK 2.0

To interact with the drone you must use the functions defined in the tello sdk.

These are some of the elementary movements:

-tello.move(dir, dist)

-tello.go(x,y,z,speed)

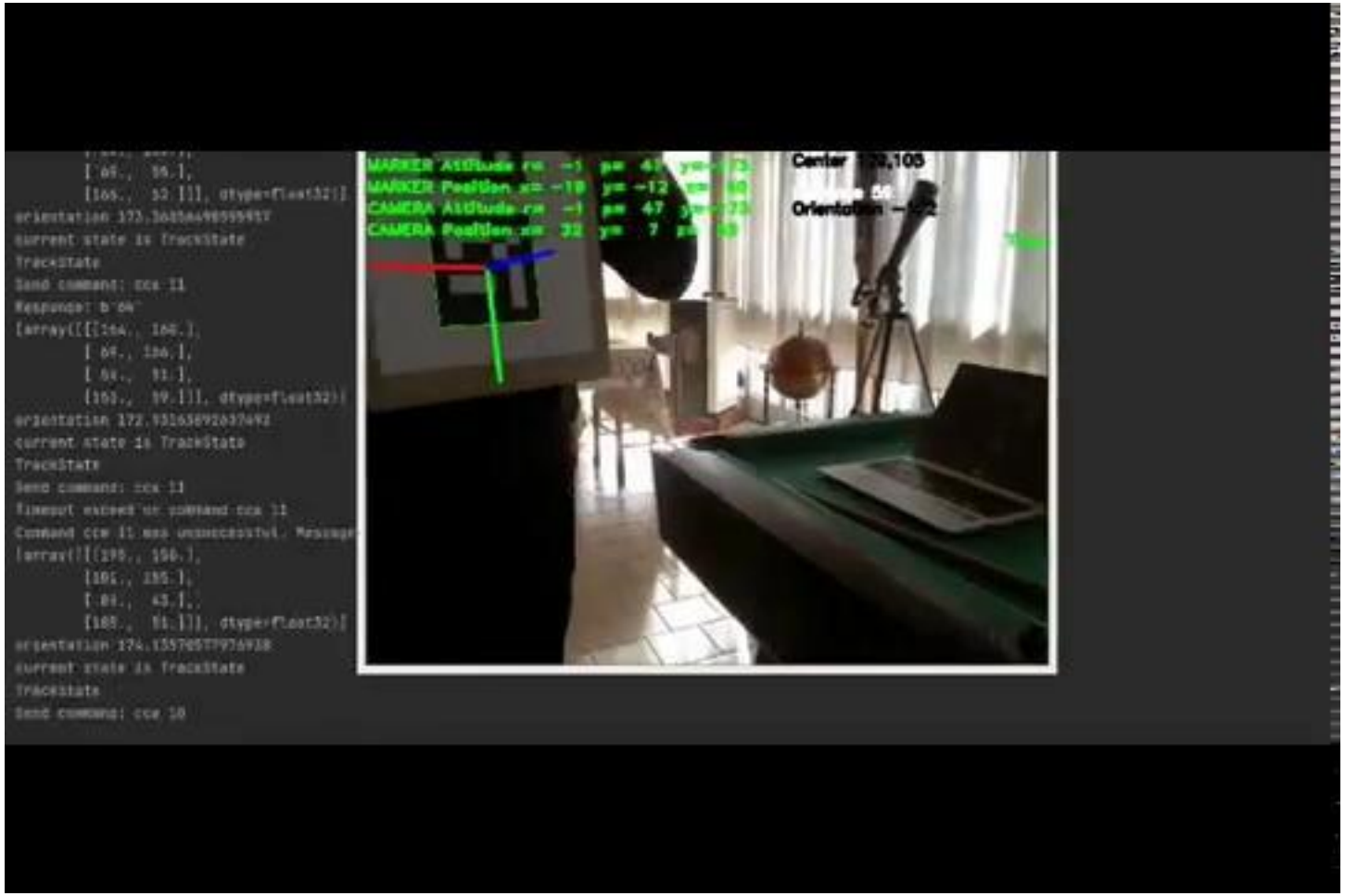
-tello.rotate_counter/clockwise(theta)

-tello.takeoff()

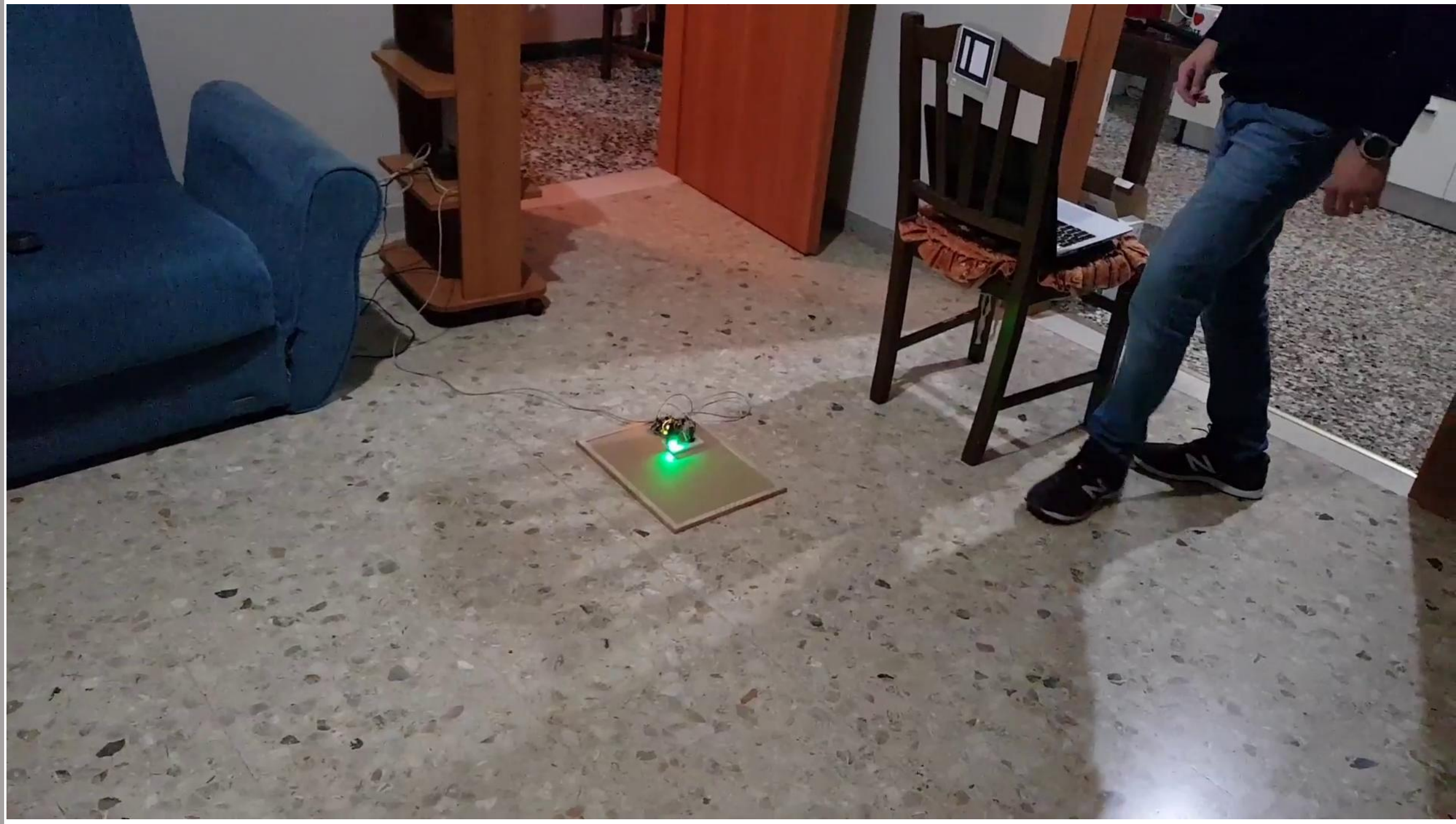
-tello.land()

with tello.get_time_of_flight() you can estimate the distance traveled. but it's not accurate for environmental agents.

Tello drone pose estimation and tracking



DEMO



Recap of future features we look forward to add

1.Alarm will be triggered **no** more by a **threshold** but using data saved as **feature vectors** and using a **labeled dataset** (with data collected through time) so that an **anomaly detection** becomes possible and the alarms system becomes more accurate

2. **One** single **account** will offer the possibility to connect to **multiple SAURUSS**

3. Drone will have **GPS** integrated so relative distances selected with a picker on the app will be replaced by current gps position(longitude,latitude) of the sensor, taken automatically on qr code scanning with the app

4. Collecting data of the single user we will be able to show him an **history** of what happened since he bought SAURUSS

5. Collecting together data from multiple SAURUSS of different users we will be able to do separate **statistics about intrusions**, statistics based on **time** and also **zone**

6. We will improve the drone movements including features of **obstacle avoidance**