ROVER SAM

Il Robot SAM (Scanner Automous Mars rover) è un prototipo dotato di:

- DC-Motors e driver di controllo;
- micro-controller arduino;
- mini-computer raspberry linux o.s.;
- ultrasonic scanner;
- IR/USB Camera (con supporto rotante);
- drone (supporto aereo);
- braccio robotico (6-axis arm);
- color-object detection (rilevamento campioni di rocce).

Il rover, mentre perlustra (con guida autonoma ed evitando gli ostacoli sul suo percorso) la superficie planetaria, effettua una scansione del territorio (inviando i dati ad un PC, che li elabora producendo una "bozza virtuale" dell'ambiente planetario). Attraverso un sistema di object detection il robot è in grado di rilevare differenti campioni di rocce (di diversa colorazione) e di afferrarli e deporli in un cestino posto sul robot. Un drone, connesso con il rover è in grado di effettuare perlustrazione aerea (a supporto del rover, nel caso in cui risultasse "bloccato tra le rocce").

Uso del software

1. Emulazione degli elementi 3d del rover

Usando un sistema linux (ad esempio ubuntu 20.04), dopo aver installato il web server apache (https://www.cyberciti.biz/faq/how-to-install-apache-on-ubuntu-20-04-lts/) creare sotto /var/www/html/ la cartella sam_rover e copiare all'interno la cartella 3d (cartella presente su drive, dopo aver de-zippato la cartella code.zip). Avviare da browser: `http://localhost/sam_rover/3d/3d_rover_elements/`. Il sistema va avviato da PC connesso a monitor touch, per consentire di esplorare i vari elementi e la relativa descrizione attraverso click sul monitor

2. Avvio sistema ultrasonico di rilevamento distanze su esp8266 e riproduzione in 3d dell'ambiente

Installazione del firmware su esp8266: connettere l'esp8266 al PC e caricare il codice presente in: https://github.com/umbertochimenti/makersproject/tree/master/SAM_rover/esp/esp_mg90s_servo_1_hcrs04_ultra_rotating_program. Al termine, testare il sistema aprendo il monitor seriale di Arduino. Si otterrà un output simile al seguente:

Ovviamente, va adattata la sezione del codice:

```
// wifi section
const char* ssid = "name";
const char* password = "password";
```

sulla base dei parametri di rete a cui l'esp effettuerà la connessione. Se il test va a buon fine, passare alla connessione web socket, tenendo conto della seguente sezione di codice:

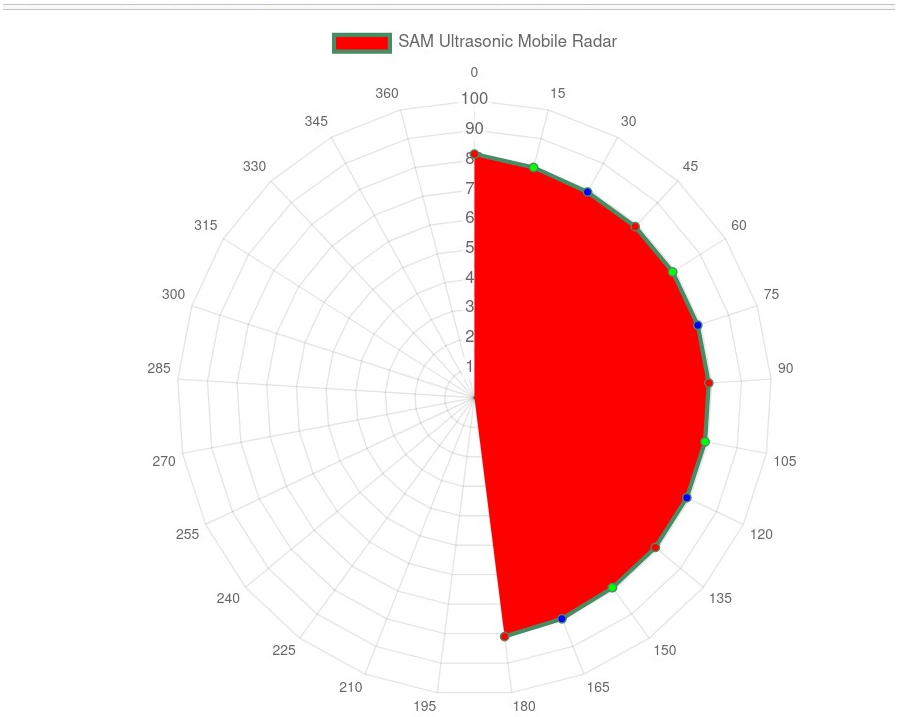
```
String websockets_server_host = "192.168.43.19";
int websockets_server_port = 9000;
```

dove va impostato ip e porta del server web socket che "gira" su PC da cui avviare il server python di connessione (codice github nel folder SAM_rover/server_python), con il comando:

```
python3 ws_server_sam_rover_radar_system.py
```

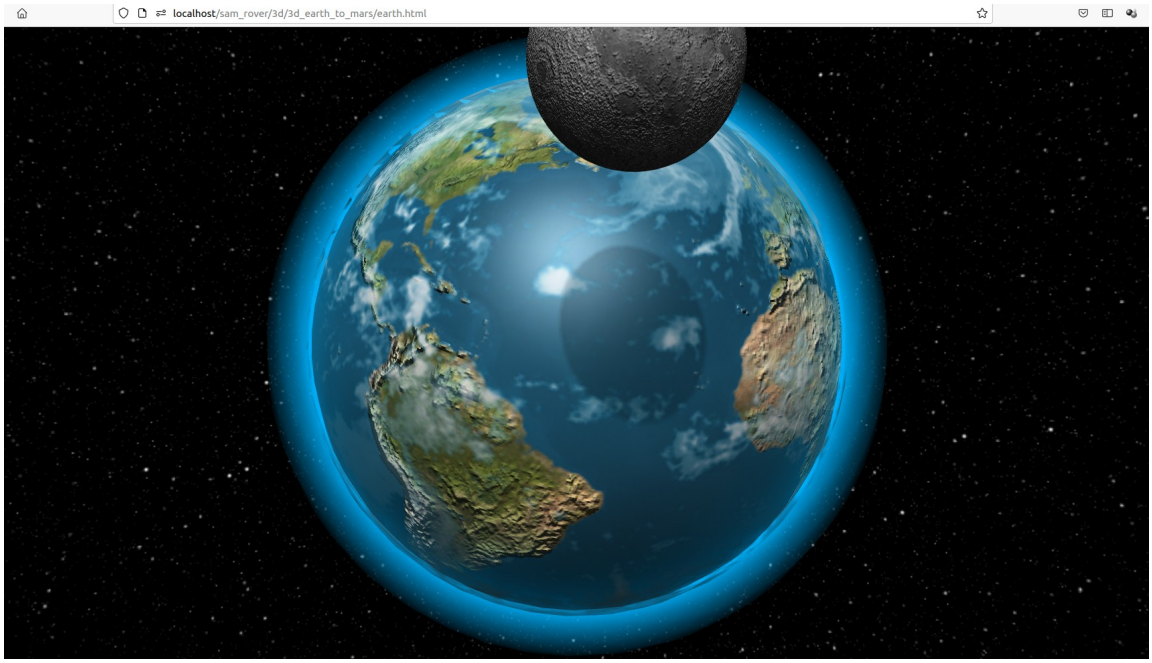
si otterrà un output simile al seguente:

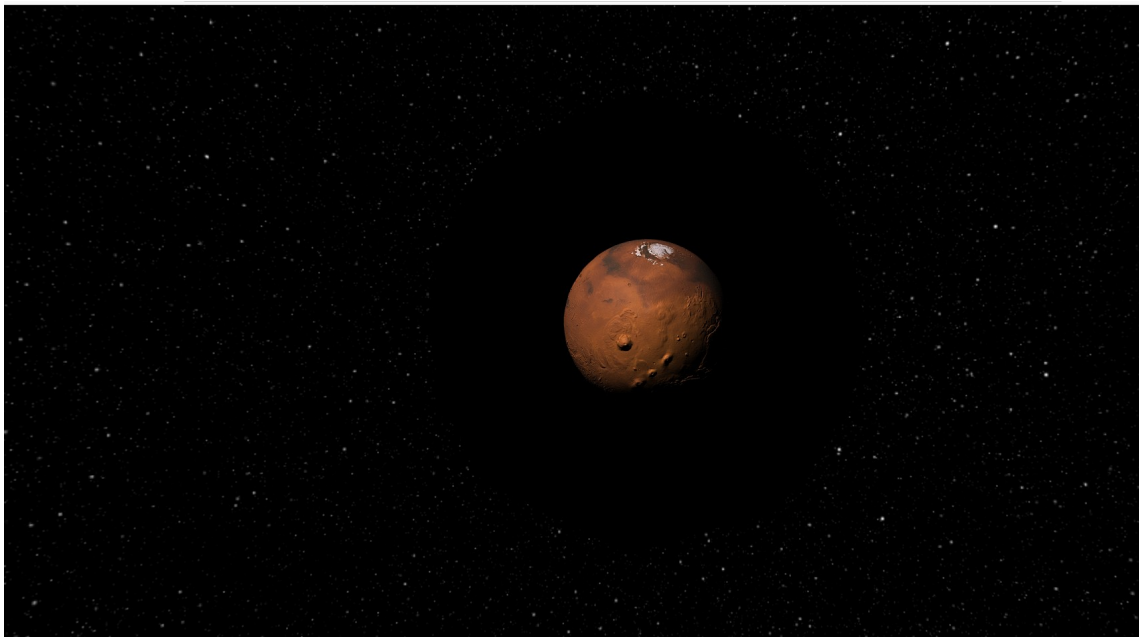
Effettuare il download del codice relativo al radar in chart js, presente al repo: https://github.com/umbertochimenti/makersproject/tree/master/SAM_rover/radar_chart_js. Lanciare da browser il codice in **`radar_chart_js/radar.html`**. Si dovrebbe ottenere un risultato simile al seguente (i dati in cm corrispondono esattamente a quanto riportato su server python):

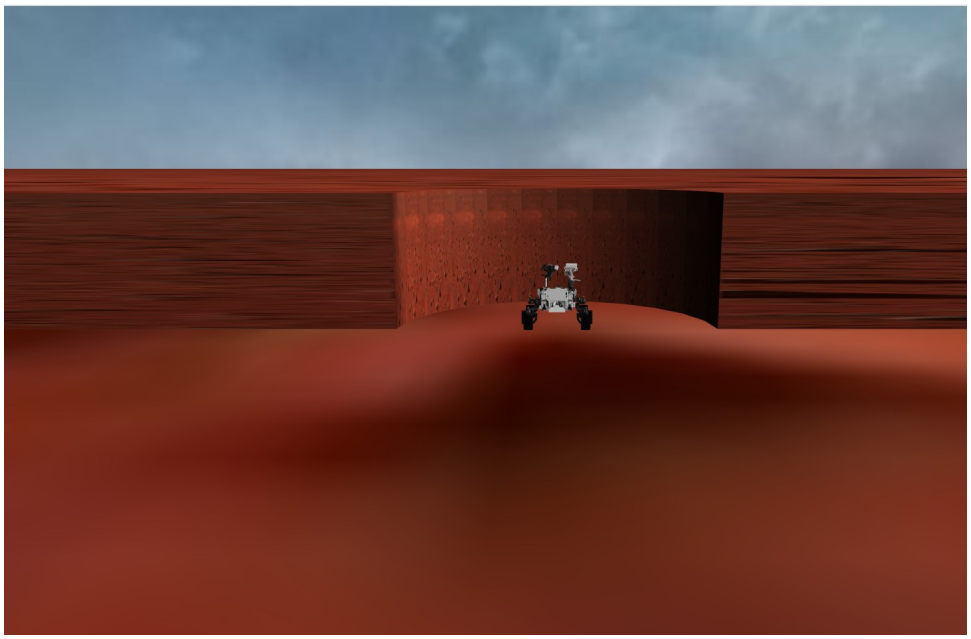


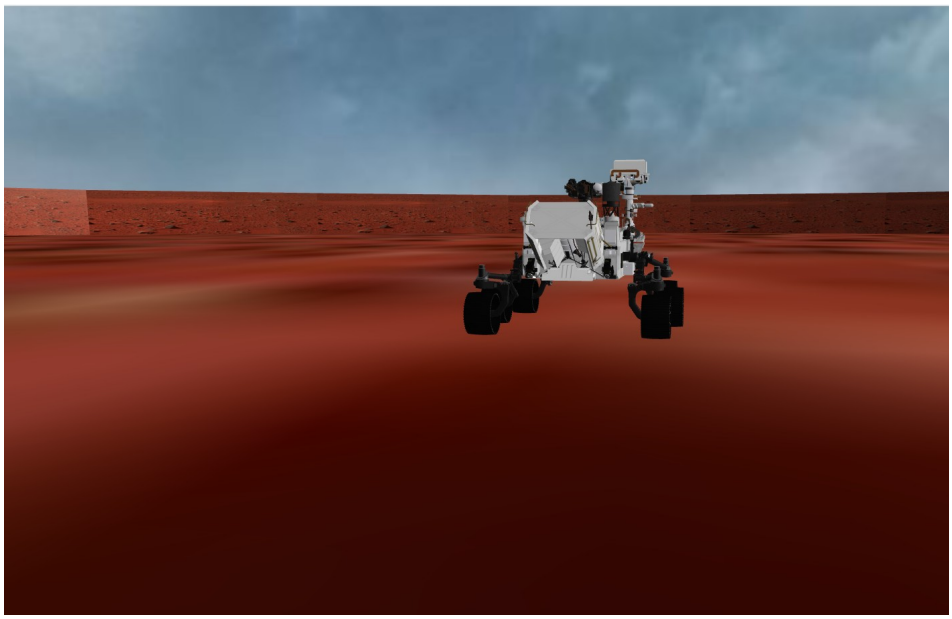
Nota: I dati del sistema ultrasonico vengono inviati dall'esp via ws-socket. Il sistema ultrasonico montato su esp va inserito sul rover e va preparato uno scenario di demo (con simulazione ostacoli e barriere laterali che delimitano l'ambiente, quest'ultimo dettaglio serve per evitare problemi sull'uso delle telecamere).

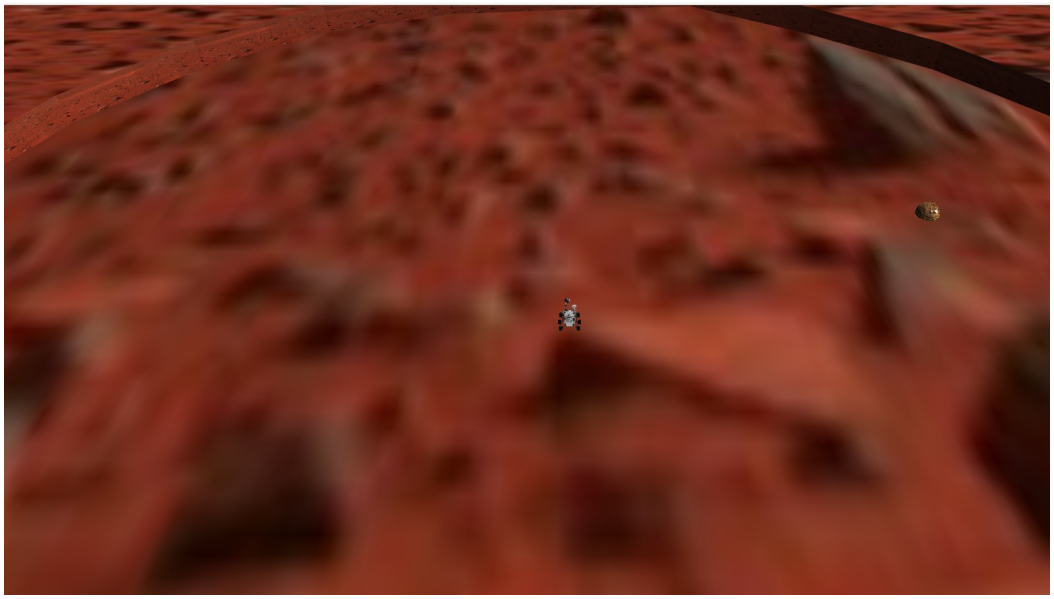
A questo punto è possibile avviare l'ambiente di emulazione 3d completo. A tal fine effettuare il download del folder: https://github.com/umbertochimenti/makersproject/tree/master/SAM_rover/3d. Il codice richiede la presenza di un web server, quindi, l'esecuzione all'interno delal cartella htdocs (per sistemi win) o /var/www/html (sistemi linux). Quindi, inserire il folder nella cartella appropriata (nel caso linux in /var/www/html/sam_rover/3d). Successivamente, lanciare da browser: **`http://localhost/sam_rover/3d/3d_earth_to_mars/earth.html`**. A questo punto sarà visibile da browser la simulazione del viaggio del rover dalla terra a marte e la successiva riproduzione 3d della superficie marziana. Con la pressione del tasto "Q" è possibile variare real-time la visuale dell'ambiente.


localhost/sam_rover/3d/3d_earth_to_mars/earth.html









3. **Avvio codice principale di controllo del rover (server su raspberry) e interfaccia di pilotaggio (client HTML5 su PC)**

Avviare su raspberry il codice del server presente su:
https://github.com/umbertochimenti/makersproject/tree/master/SAM_rover/rover_main/server

con il comando:

python3 ws_server.py

se il sistema è impostato in modo corretto, si otterrà il seguente output:

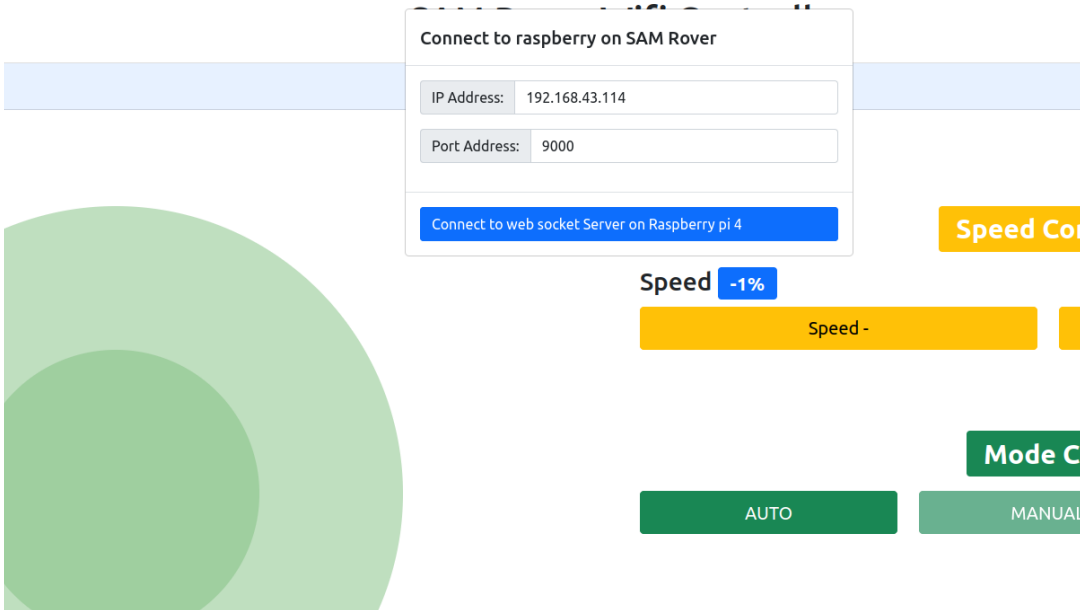
```
pi@raspberrypi:~/server $ python3 ws_server.py
[MAIN] INFO: start wifi robot control program!
[INFO] setup done!
[INFO] l293d setup OK!
[INFO] Start dht11 sensor thread manage!
[INFO] Start hole detect thread!
[INFO] setup done!
set servo pin: 0 at angle: 90
set servo pin: 1 at angle: 90
set servo pin: 2 at angle: 90
set servo pin: 3 at angle: 90
set servo pin: 4 at angle: 90
set servo pin: 5 at angle: 90
[INFO] Setup robot 6-axis arm complete!
[MAIN] INFO: Web Sockets server listen on 9000
```

Se dovesse verificarsi l'errore: *AttributeError: module 'board' has no attribute 'SCL'* eseguire le seguente procedura:

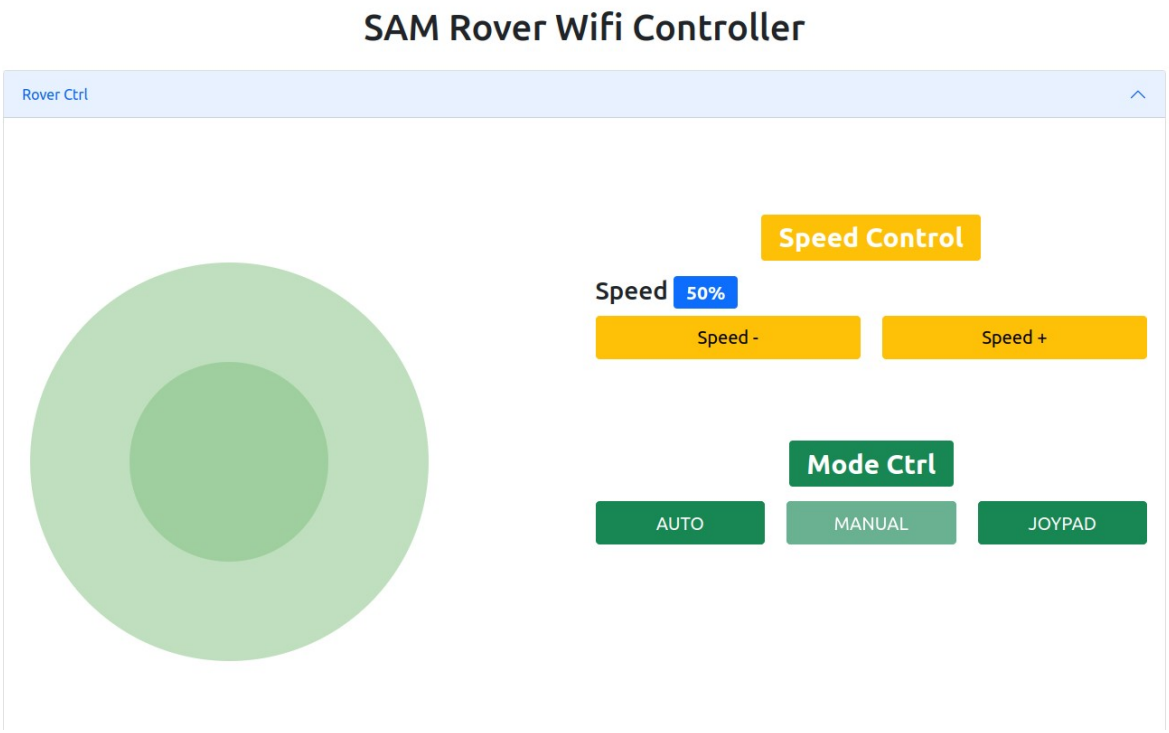
```
pip3 uninstall board
pip3 install adafruit-blinka
```

Sul PC, avviare il sistema client, lanciando da chrome il file robot_controller.html, dopo aver caricato sul file system l'intero folder presente in:
https://github.com/umbertochimenti/makersproject/tree/master/SAM_rover/rover_main/client

Su browser verrà renderizzata la seguente schermata di main con una gui modale di connessione:



Di default, in fase di test è stato impostato il sistema server su IP 192.168.43.114, in ascolto sulla porta 9000. Ovviamente, l'indirizzo IP rappresenta un parametro da modificare sulla base dell'IP del raspberry (ATTENZIONE: si rammenta che il PC e il raspberry devono essere in esecuzione sulla stessa rete LAN, ad esempio, attraverso connessione via router wifi dello smartphone. Il raspberry va impostato in modo tale che in fase di avvio rilevi in modo automatico la rete wifi dello smartphone. Si suggerisce di impostare i parametri di connessione per raspberry nel file wpa_supplicant.conf). Dopo aver premuto sul tasto blu di connessione (tasto: connect to web socket Server on raspberry pi 4), comparirà il messaggio INFO: connection ok! (oppure il messaggio ERROR: connection error! Nel caso di mancata connessione) e comparirà la seguente interfaccia:



A questo punto è possibile usare l'interfaccia di pilotaggio nelle componenti:

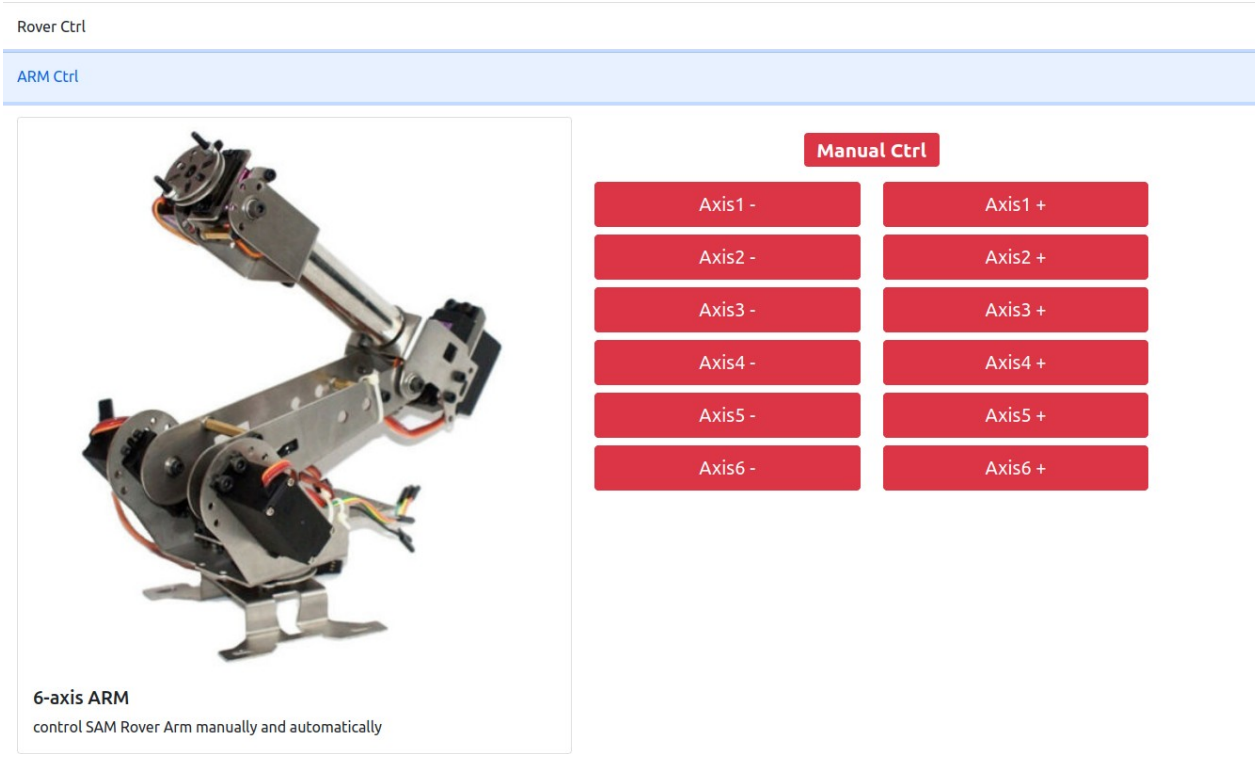
- virtual joypad (up, back, left, right);
- Speed Control: Speed+, Speed-;
- Mode Ctrl: tasti MANUAL e JOYPAD;

e monitorare i valori di temperatura ed umidità (nella sezione T/H Sensor Update).

ATTENZIONE: il tasto AUTO (di Mode Ctrl) non è ancora utilizzabile, dato che il valore dei sensori ultrasonici e dei sensori IR non compare a video (infatti le sezioni Ultrasonic Sensors Update e Hole Sensors Update non riportano valori numerici, ma solo le stringhe "–"). Dopo aver effettuato alcune prove di movimentazione e variazione di velocità, passare al pilotaggio da Joypad. Connettere un joypad USB al PC, simile ai seguenti:

Successivamente premere il tasto JOYPAD nella sezione Mode Ctrl per pilotare il rover tramite joypad. In tal caso, si ricordi che i tasti direzionali effettuano la movimentazione up, back, left, right e con il tasto rosso lo stop_motor.

Se tutti i test effettuati risultano ok, passare alla schermata di gestione dell'ARM, switchando sulla sezione ARM Ctrl:



e testando tutti i 12 tasti relativi alla movimentazione JOG dell'ARM (movimenti manuali). Se tutti i movimenti avvengono in modo corretto, il test si può considerare concluso con esito positivo.

4. Avvio del server di gestione dei sensori (in node.js) su su raspberry e gestione delle ulteriori funzionalità di interfaccia di pilotaggio

Si rammenta che il sistema di gestione dei sensori richiede la parte di sensoristica connessa su Arduino (sensori ultrasonici, IR e giroscopio/accelerometro). Arduino va connesso ad una porta USB del raspberry. Su Arduino va caricato il firmware firmata (seguire la guida: http://johnny-five.io/platform-support/). Installare il PingFirmata.ino (https://gist.githubusercontent.com/rwaldron/0519fcd5c48bfe43b827/raw/f17fb09b92ed04722953823d9416649ff380c35b/PingFirmata.ino). Su Raspberry va installato node js ed npm (vedi Appendice B) e la libreria Johnny-Five e websocket con i comandi:

```
npm install johnny-five
npm install websocket
```

Effettuare il test del blink in node js. Si potrebbe optare per Arduino Mega, piuttosto che per Arduino UNO (in modo da avere a disposizione una board con un numero maggiore di pin per eventuali sviluppi futuri).

Hello World!

Microcontrollers and SoC platforms like to say "Hello World" with a simple blinking LED; the following demonstrates how to do this with the Johnny-Five framework.

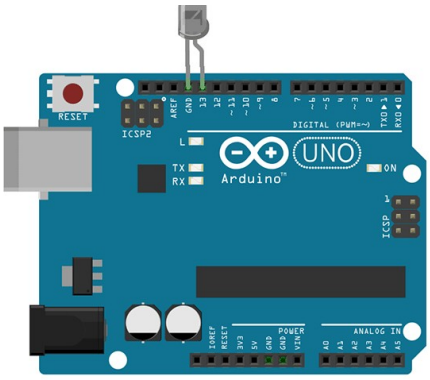
1. Install Node.js (Prefer LTS!).
2. Setup your board.
3. Get Johnny-Five: `npm install johnny-five` *
4. Run your program! `node blink.js`

Arduino | Tessel 2 | Raspberry Pi | Intel Edison | Particle Photon | See All...

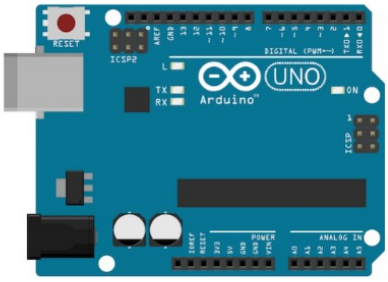
```
var five = require("johnny-five");
var board = new five.Board();

board.on("ready", function() {
  var led = new five.Led(13);
  led.blink(500);
});
```

Browse more examples

* There are likely to be other steps involved in preparing your target platform, the best place to start is by looking at Platform Support.



Arduino Uno



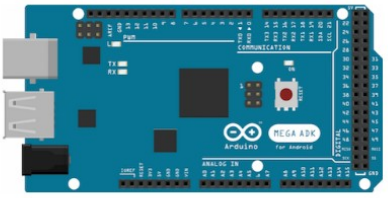
Environment

- Firmware/Runtime: StandardFirmataPlus (additional instructions)
- The JavaScript program is executed on a **host** machine that runs Node.js. The program transmits basic IO instructions to the board via **usb serial**, which acts as a **thin client**. Requires tethering.

Platform Specific

- Supports the `PING_READ` extension, when used with `PingFirmata`.
- Supports the `STEPPER_*` extensions when used with `AdvancedFirmata` or `ConfigurableFirmata`.

Arduino Mega



Environment

- Firmware/Runtime: StandardFirmataPlus (additional instructions)
- The JavaScript program is executed on a **host** machine that runs Node.js. The program transmits basic IO instructions to the board via **usb serial**, which acts as a **thin client**. Requires tethering.

Platform Specific

- Supports the `PING_READ` extension, when used with `PingFirmata`.
- Supports the `STEPPER_*` extensions when used with `AdvancedFirmata` or `ConfigurableFirmata`.

Successivamente, lanciare su raspberry il codice del server node js, presente al repo: https://github.com/umbertochimenti/makersproject/tree/master/SAM_rover/ultrasonic_and_gyro_node_js_ctrl/server

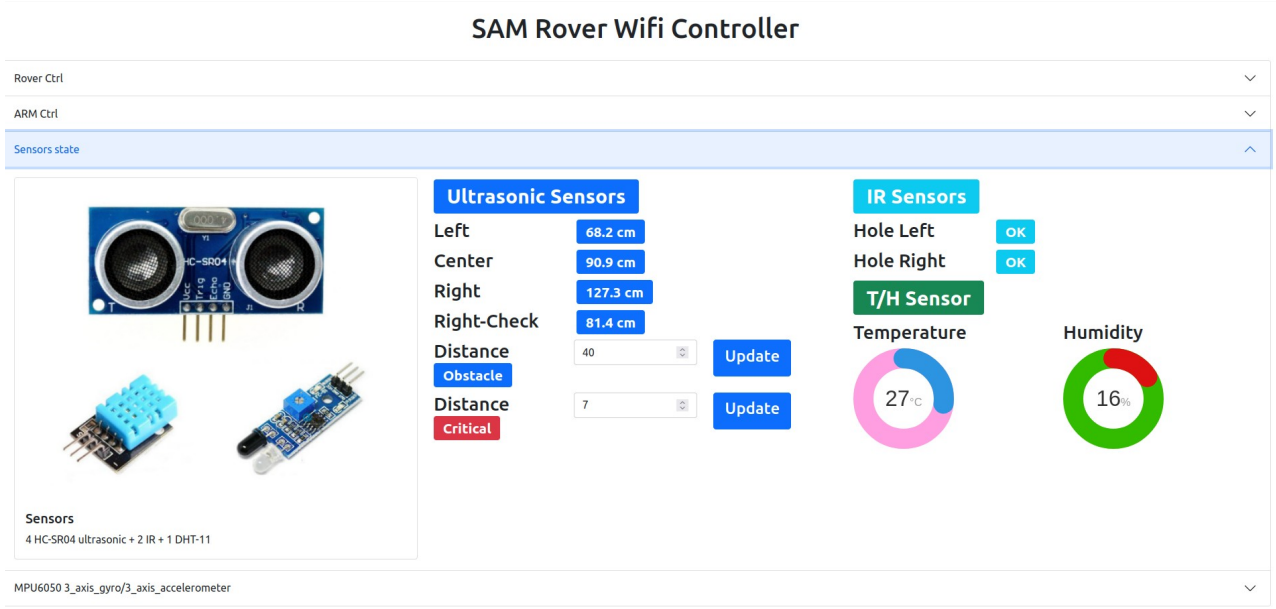
con il comando:

node rover_sensor_ctrl.js

se il server funziona correttamente si visualizzerà il seguente output su terminale:

```
pi@raspberrypi:~/server_node $ node rover_sensors_ctrl.js
1631800426790 Connected /dev/ttyACM0
[INFO] server ws listen on 4500!
1631800430758 Repl Initialized
>>
```

A questo punto bisogna riconnettere la schermata principale di pilotaggio (effettuando refresh da browser). Se il server node funziona correttamente ed è in grado di rilevare i dati dei sensori, è possibile monitorare i valori dei sensori attraverso la schermata sensors state ed MPU6050:



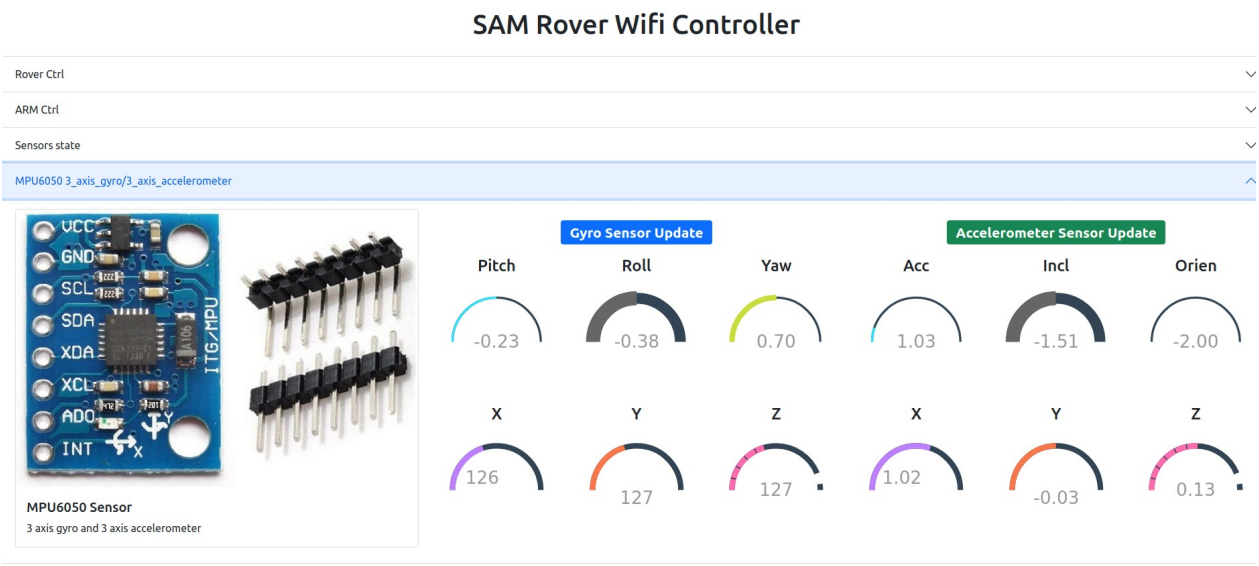
La schermata sensors state mostra:

- i valori dei sensori ultrasonici, che rilevano le distanze dell'ambiente dal robot (in particolare la distanza left, right, center e il valore di right-check che serve per ottimizzare le movimentazioni in modalità automatica);
- i valori dei sensori IR con l'indicazione dello stato OK per l'hole left e right;
- i valori rilevati dal sensore di temperatura/umidità DHT11.

Inoltre, consente di effettuare il setting del valore di distance Obstacle e Distance Critical. Entrambi i valori sono utilizzati per la modalità di pilotaggio automatico del rover, in particolare:

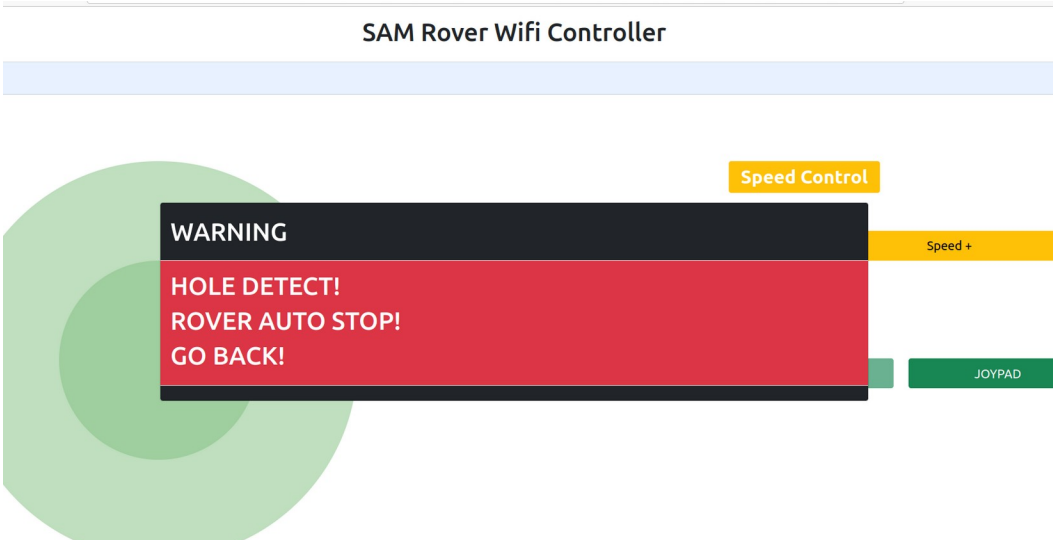
- distance obstacle: rappresenta la distanza rilevata la quale il rover effettua una sterzata per evitare l'ostacolo;
- distance critical: rappresenta la distanza rilevata la quale il rover effettua una manovra di back rapido per evitare "scontri con ostacoli ravvicinati" non identificati in anticipo.

La seguente schermata rappresenta i numerosi dati estratti dal modulo MPU6050 (dati al momento solo visualizzati, ma torneranno utili per sviluppi futuri).



Per quanto riguarda il giroscopio/accelerometro, effettuare un test manuale, muovendo l'MPU6050 e osservando con attenzione la variazione dei parametri giroscopici:

A questo punto è possibile testare anche la funzionalità di pilotaggio automatico, premendo il tasto AUTO nella sezione Mode Ctrl della schermata principale. È possibile testare il funzionamento dei sensori IR "rileva buche", spostando manualmente il rover su un piano rialzato in prossimità del bordo. In automatico il rover si fermerà in stato di emergenza e comparirà su interfaccia principale il seguente messaggio di alert:



5. Web streaming telecamera posta su braccio robotico

E' stata installata una telecamera "su cavo flessibile" terminante sulla pinza dell'arm, da utilizzare per il monitoraggio della fase di "acchiappo" dell'elemento. La gestione della stessa avviene in python con libreria Flask. Caricare sul raspberry il codice presente in https://github.com/umbertochimenti/makersproject/tree/master/SAM_rover/web_streaming_arm. Il codice va modificato nel file app.py, in particolare l'istruzione:

app.run(host='192.168.43.114', port=5000, threaded=True)

va, ovviamente, modificata con l'IP del raspberry. Successivamente, avviare l'applicazione con il comando:

```
python3 app.py
```

l'output del programma è simile al seguente:

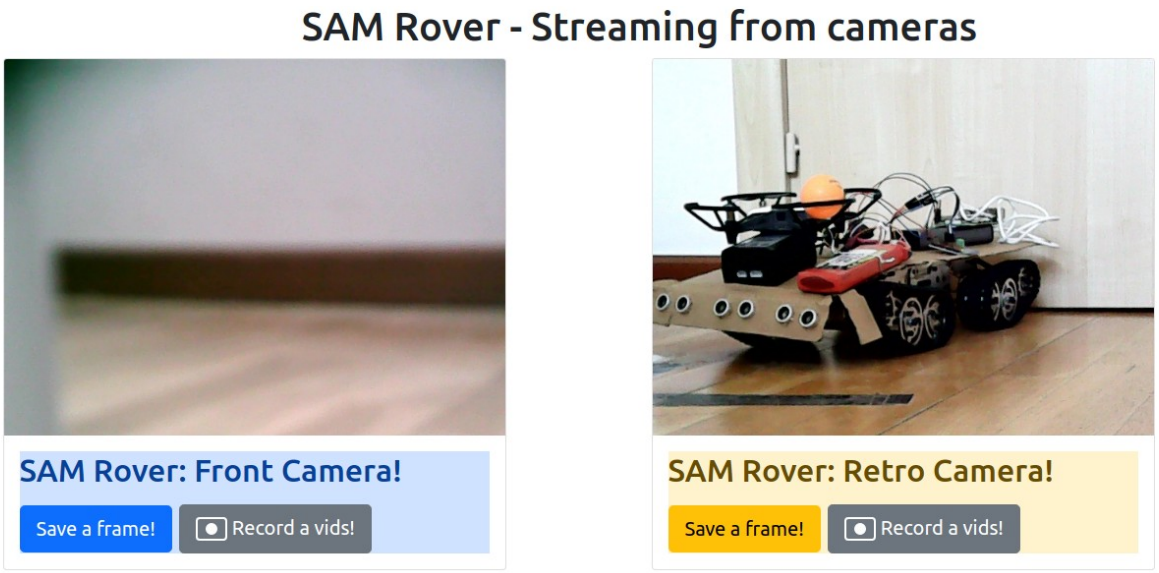
```
[INFO] start to capture camera from OpenCV ...
 * Serving Flask app "app" (lazy loading)
 * Environment: production
    WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://192.168.43.114:5000/ (Press CTRL+C to quit)
```

In questo caso, basterà aprire il browser sull'URL http://192.168.43.114:5000 per poter visualizzare l'applicazione in Flask che effettua web streaming:

è possibile salvare il frame (le immagini sono salvate sul raspberry nella cartella frames) e registrare un video (i video sono salvati sul raspberry nella cartella vids)

6. Web streaming con doppia telecamera (front e retro camera)

Il rover è dotato di due usb-camera (frontale e retro) per effettuare streaming con registrazione video e salvataggio frames. Il sistema è installato su un ulteriore raspberry, considerato che il precedente è già ampiamente impiegato per la gestione di sensoristisca, movimentazione rover e arm e telecamera su braccio robotico. Il sistema è del tutto identico al sistema streaming per arm, ma con la gestione di una doppia telecamera. Il codice è presente al link: https://github.com/umbertochimenti/makersproject/tree/master/SAM_rover/web_streaming



7. Riconoscimento oggetti in HSV via streamign RTCP

Sul rover è stato inglobato un sistema di riconoscimento per colori, al momento con maschere OpenCV in grado di filtrare oggetti rossi e blu. Il sistema è avviato, utilizzando la camera connessa al primo raspberry (dove è collocata anche la camera per il braccio robotico, vedi punto 5). nello specifico il codice è presente al repo github:

è necessario installare su raspberry e su PC la libreria di streaming RTCP, con il seguente comando:

```
pip3 install imagezmq
pip3 install imutils
```

e lanciare il comando:

```
python3 stream_client.py
```

al che potrebbe verificarsi l'errore:

```
VIDEOIO ERROR: V4L: can't open camera by index 3
```

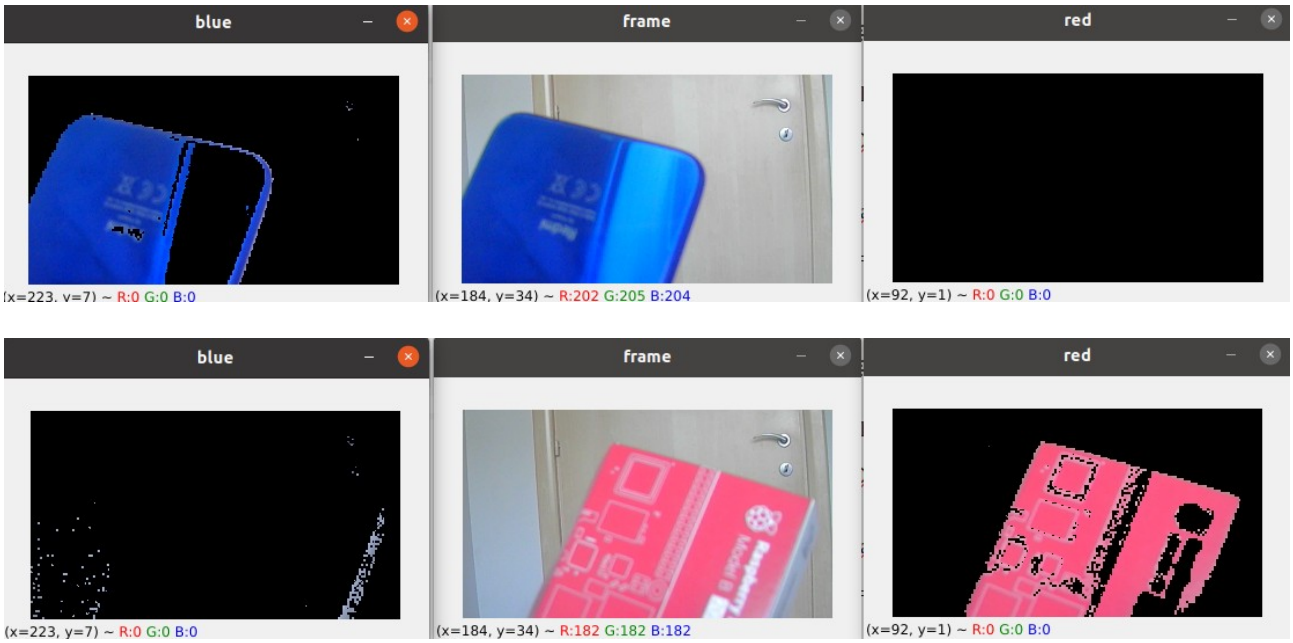
che indicherebbe la necessità di settare il corretto indice della camera all'interno del file stream_client.py:

```
vs = VideoStream(src=2, resolution=(160, 120)).start()
```

con il comando: ***ls /dev/video**** si possono notare i valori numerici associati alle camere inserite. Quindi, estraendo e reinserendo la telecamera si può notare il file imputato alla camera connessa e variare l'indice sul codice. Successivamente sul PC avviare la parte server, con il comando:

```
python3 stream_server_hsv_mask.py
```

ottenendo un output simile ai seguenti:



8. Avvio openCV tracking rover control system

documentazione in elaborazione

9. Avvio gestione drone Ryze Tello

documentazione in elaborazione

10. Avvio gestione riconoscimento oggetti con rete neurale

documentazione in elaborazione

Appendice A: Installazione del sistema operativo su SD-Card

Link Guide:

- https://www.raspberrypi.org/documentation/installation/installing-images/linux.md
- https://www.instructables.com/How-to-Install-RASPBERRY-PI-OS-on-MicroSD-Card-Usi/
- https://www.instructables.com/Howto-Installing-Raspberry-PI-4-Headless-VNC-With-/

Esempio di installazione con Rasp OS (nuova versione)

Effettuare il download con click su Raspberry Pi OS with desktop dal link: https://www.raspberrypi.org/software/operating-systems/. De-zippare il file scaricato (che avrà un nome simile al seguente: 2021-05-07-raspios-buster-armhf.zip), ottenendo il relativo file .img (ad esempio: 2021-05-07-raspios-buster-armhf.img). Inserire una sd-card e lanciare il seguente comando (effettuare la deep-copy del file nella sd-card):

```
dd bs=4M if=2021-05-07-raspios-buster-armhf.img of=/dev/sdb status=progress conv=fsync
```

si tenga presente che il parametro of=/dev/sdb dipende dal punto nel file system associato alla sd-card. Successivamente, estrarre la sd-card e reinserirla. **All'interno del folder boot nell'sd-card** (ad esempio in /media/sd-card/boot), creare il file vuoto ssh, con il comando:

```
touch ssh
```

e il file wpa_supplicant.conf, contenente le seguenti righe di configurazione:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=IT

network={
ssid="your_network_ssid"
psk="your_network_password"
key_mgmt=WPA-PSK
}
```

Successivamente, inserire la sd-card nell'apposito alloggiamento del raspberry e alimentare lo stesso. Verificare che avvenga la connessione wireless con lo smartphone. Segnare l'indirizzo IP (che chiamiamo rasp_ip_address) del raspberry ed effettuare un comando ping da PC su raspberry. Se è tutto ok, avviare una connessione ssh con raspberry, attraverso il comando:

```
ssh pi@rasp_ip_address
```

o anche:

```
ssh -X pi@rasp_ip_address
```

Il sistema vi richiederà una password di accesso, che di default è ***raspberry***.

Appendice B: configurazione iniziale

Dopo aver avviato Raspberry con connessione SSH, configurare in ON da raspi-config sia l'interfaccia SPI, sia l'interfaccia I2C. Avviare da terminale:

```
sudo apt-get update
sudo apt-get upgrade
```

Se si utilizza una delle ultime versioni di rasp OS, lanciare i comandi, prima di avviare il punto 3:

```
pip3 uninstall board
pip3 install adafruit-blinka
```

Installare le seguenti librerie:

```
pip3 install SimpleWebSocketServer
pip3 install adafruit-circuitpython-pca9685
pip3 install adafruit-circuitpython-servokit
pip3 install adafruit-circuitpython-dht
sudo apt-get install libgpiod2
```

Installazione nodejs:

```
sudo apt-get update
sudo apt-get install nodejs
sudo apt-get install npm
```

Appendice C: Installazione OpenCV e Python Flask

Inizialmente è necessario installare la libreria openCV che consente di catturare i frame dalla usb camera e di effettuare operazioni di image processing. La procedura richiede un po' di tempo, ma essenzialmente si tratta di lanciare comandi di installazione da terminale linux sotto raspberry.

Vanno installate una serie di librerie con i seguenti comandi (si consiglia di lanciare un comando per volta e attendere l'installazione delle librerie):

https://www.pyimagesearch.com/2019/09/16/install-opencv-4-on-raspberry-pi-4-and-raspbian-buster/

```
sudo apt-get install build-essential cmake pkg-config
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
sudo apt-get install libfontconfig1-dev libcairo2-dev
sudo apt-get install libgdk-pixbuf2.0-dev libpango1.0-dev
sudo apt-get install libgtk2.0-dev libgtk-3-dev
sudo apt-get install libatlas-base-dev gfortran
sudo apt-get install libhdf5-dev libhdf5-serial-dev libhdf5-103
sudo apt-get install libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5
sudo apt-get install python3-dev
```

inoltre installare:

```
sudo apt-get install libilmbase-dev
sudo apt-get install libopenexr-dev
sudo apt-get install libgstreamer1.0-dev
```

quindi installare opencv per python3 con il tool pip3:

```
pip3 install opencv-contrib-python==4.1.0.25
```

al termine installare la libreria flask per creare una web app in python per lo streaming da usb camera:

```
pip3 install flask
```

Appendice D: Connessioni hardware

Arduino è connesso via USB al raspberry pi 4 master.
Ad Arduino sono connessi:

- 4 sensori ultrasonici:
  - sensore left: al pin 7;
  - sensore center: al pin 6;
  - sendore right: al pin 5;
  - sensore right-check: al pin 4.
- 2 sensori IR (usati come sensori "rileva buche"):
  - IR_left: al pin 2;
  - IR_right: al pin 3.
- 1 sensore MPU5060 (giro/accel):
  - pin SCL del sensore: al pin A5;
  - pin SDA del sensore: al pin A4.

Al raspberry master è connesso:

- 1 sensore DHT11: al pin GPIO 17;
- 1 modulo pca9685 (controller a 16 cnali PWM):
  - pin SCL del modulo: al pin GPIO 2;

- pin SDA del modulo: al pin GPIO 3.