braccio$_{c}amai$

Release 0.1

Dec 16, 2019

Contents

Braccio CamAI is a low-cost robot arm powered by computer vision and artificial intelligence, perfect for experimenting and DIY projects.

Instruccions for building and installing the project are available, as well as step-by-step tutorials for face and object detection and tracking.

Source code is available on:

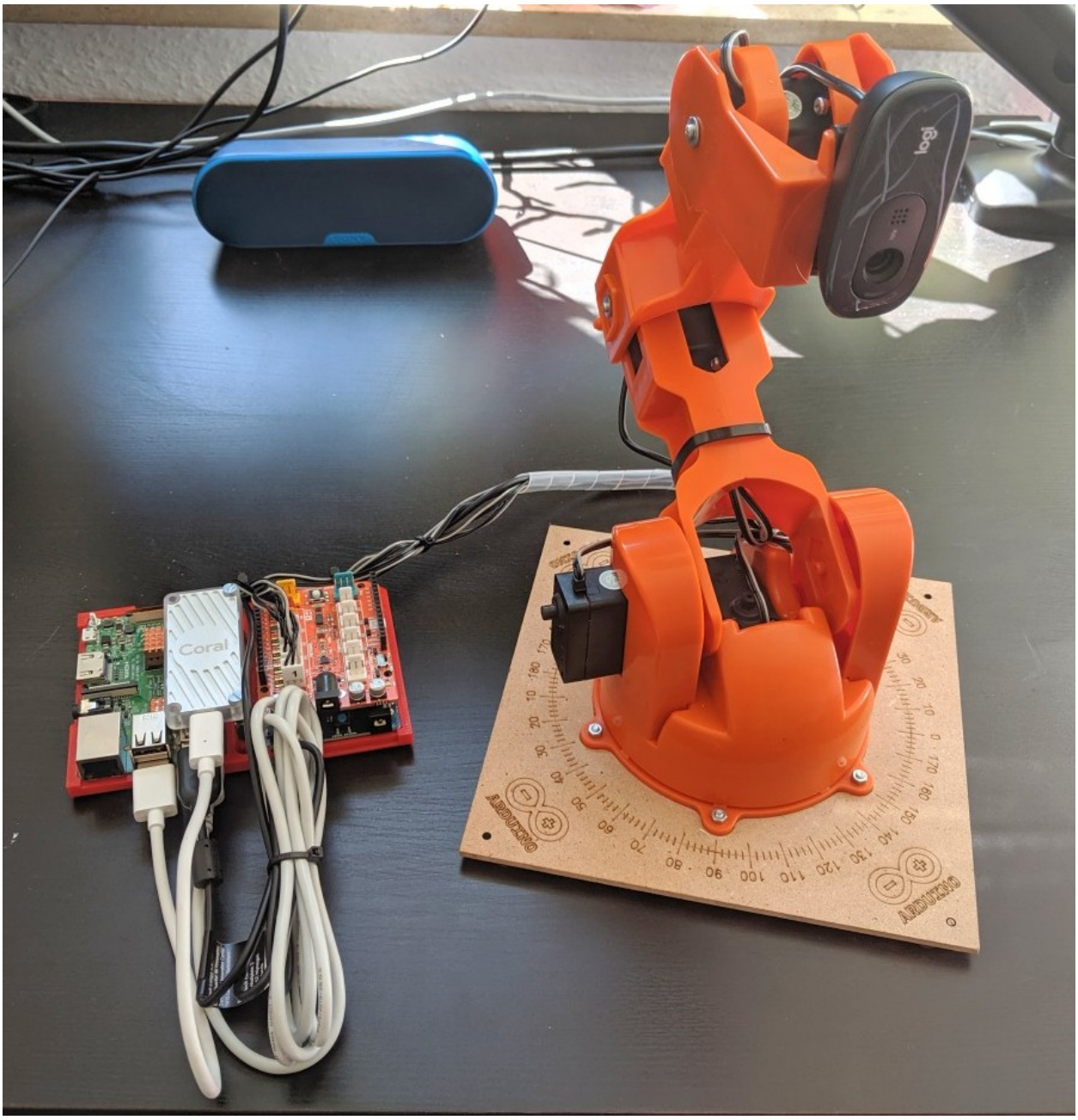
https://github.com/francisc0garcia/braccio_camai

CHAPTER 1

Contents:

1.1 Project's description

Using an Arduino Tinkerkit Braccio Robot, Raspberry Pi 3 and a Google Coral USB Accelerator, it allows you to actively track and follow more that 90 different type of objects autonomously. It is also fully integrated in ROS (Robot Operating System) using MoveIt or a simple Joint interface, and uses state-of-the-art algorithms for object detection or classification powered by TensorFlow Lite.



Detailed instructions related to robot-arm assembly, connections and electronics, software and tutorials are given in the following sections.

1.2 Mechanical assembly

1.2.1 Robot arm: Tinkerkit Braccio Robot

One of the main mechanical component of Braccio CamAI is the robot arm: Tinkerkit Braccio Robot. It has 6 degrees of freedom (DOF), and uses servo-motors controlled by arduino using PWM (Pulse width modulation). For simplicity, we have excluded two components (Gripper and last Servo-motor) from the original model. Resulting in a 4-DOF arm. Notice that the URDF models contains all 6-DOF, in case you want to keep them all and build new applications.

Detailed instructions for building the arm are available: Tinkerkit Braccio Assembly or Tinkerkit video tutorial.

Important: Since the servo-motors do not provide any joint-position feedback, it is important to properly calibrate the arm. Arduino provides the script testBraccio90.ino that moves the robot arm to the straight position, please check that the arm is straight after running it.

1.2.2 Camera

Instead of using the last servo-motor and gripper, we have installed a Logitech C270 camera. It allows us to capture images directly on the Robot End-effector.

To facilitate the camera installation, a 3D-printed base was designed for this project: Logitech c270 base model. We attached the camera using the 3D-printed model and some additional cable ties.

Remember to pass the USB camera cable inside the inner holes of the robot arm, because the Connector is to big for for passing it through after assembly.

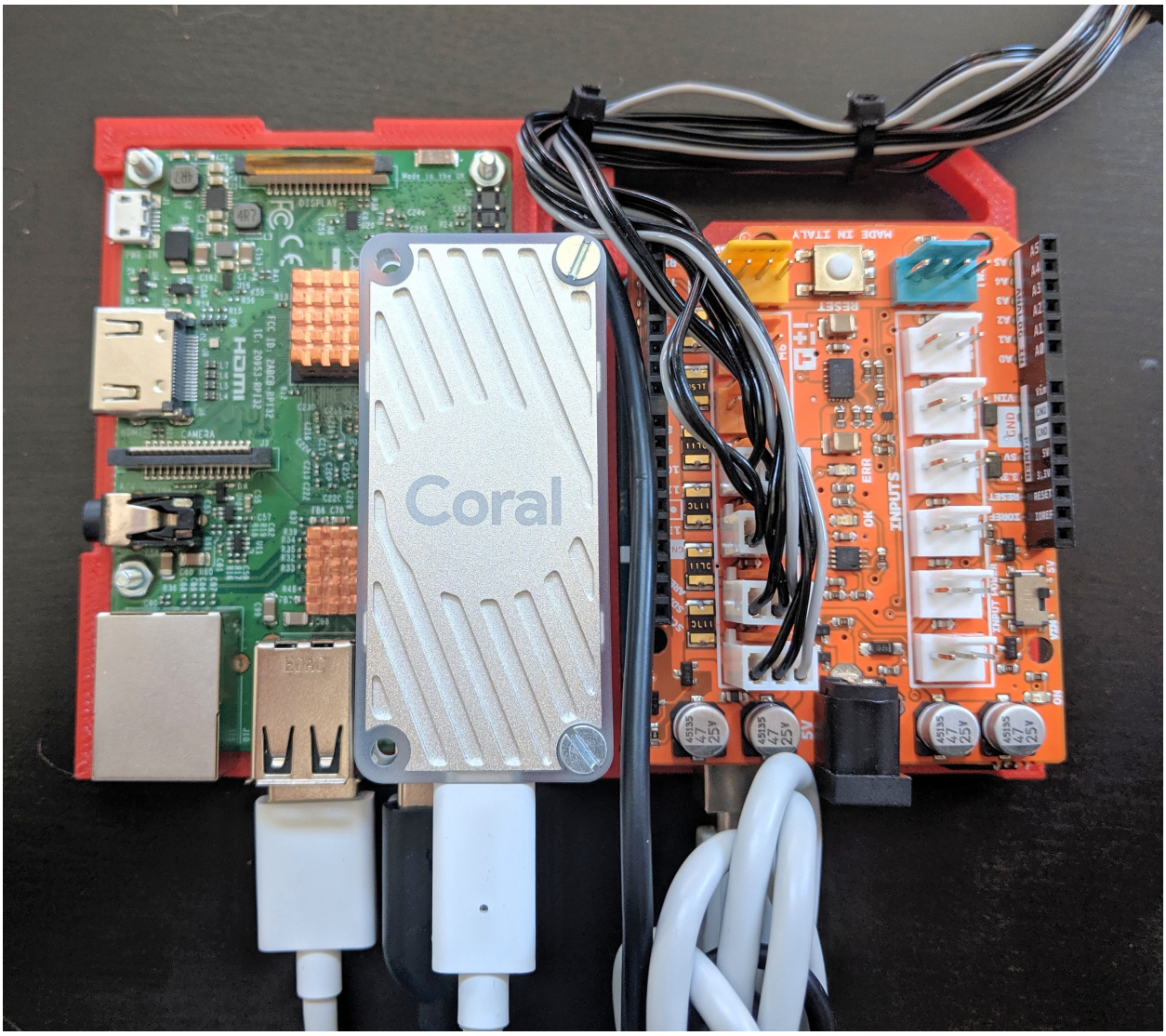
Ideally, any USB webcam can be used for the project, as long as it has linux-compatible drivers. You can check some of there here: linux-compatible UVC cameras.

1.2.3 (optional) 3D-printed base for Raspberry + Arduino

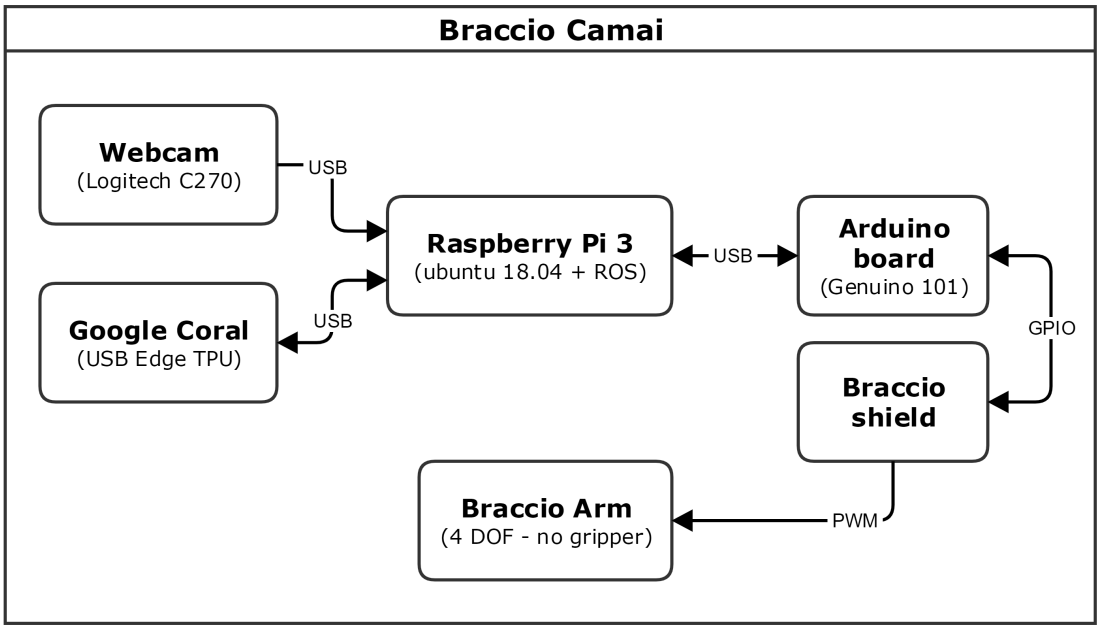
We use a 3D-printed Arduino Raspberry Pi Mount for attaching all electronic components.

1.3 Electronics

The main electronic components are detailed below:



The corresponding Connection diagram of Braccio CamAI:



All main components are described as follows:

1.3.1 Raspberry Pi 3 Model B

We use the RP3 as main processor, in charge of receiving images from the camera, process it using a Google coral USB accelerator and generating the corresponding Joint commands for the arm. It runs Ubuntu 18.04 and ROS Melodic.

Currently, we are using a *Raspberry Pi 3 Model B V1.2*, but the software and electronic connections are compatible with the newer version *RP3 Model B+*, as well as previous version *Raspberry Pi 2*.

Important: The RP3 might get really hot. At full operation, Braccio CamAI uses approximately 65% of all CPU power. If you plan to have a long run-time operation, we suggest you to install an additional fan or cooling system.

We use a standard 2 Amp - 5 Volt power adapter for powering the Raspberry Pi 3.

A ready-to-use image will be available soon, manual instruction for installing ROS and melodic on RP3 are given as follows:

- Ubuntu 18.04 on RP3: Ubuntu-18-04-RaspberryPI.
- ROS Melodic on RP3: ROS Melodic Installation.

1.3.2 Arduino Board

For controlling the Braccio Arm, we use a Genuino 101, but any Arduino-compatible board should works as well.

The firmware can be directly installed using the Arduino IDE: *Tools->Manage libraries* and search for *Braccio*. Please install both **Braccio** and **BraccioRobot**.

1.3.3 Arduino compatible Shield

The Braccio robot arm already includes an Arduino-shield for sending PWM commands to all servo-motors. It must be separately powered using its own power plug, the power adapter is already included on the Braccio kit. We left

intentionally unconnected the motors **M5** and **M6**, since we do not want to use the gripper + last servo motor.

1.3.4 Google Coral USB Accelerator

To perform object detection and classification, we have attached a Google coral USB Accelerator to the RP3.

It has ready-to-use instructions described here: Google Coral USB Accelerator get-started guide.

During installation, please remember to disable the **maximum operating frequency** or install an additional fan, because the board itself might get really hot.

1.4 How to install Braccio CamAI package

The easiest way to use the package is using our RPI3 image (Available soon!).

If you want to compile the package manually, or use a different processing unit (other than RPI), please follow the next steps:

1.4.1 Pre-requirements

We use Ubuntu 18.04 running ROS melodic on our Raspberry Pi. Instructions for installing both are given in:

- Ubuntu 18.04 on RP3: Ubuntu-18-04-RaspberryPI.
- ROS Melodic on RP3: ROS Melodic Installation.

Braccio CamAI package should be compatible with ROS Kinetic, but we haven't tested jet.

1.4.2 Compilation

- Install dependencies:

```
sudo apt install ros-melodic-moveit-core ros-melodic-moveit-ros-perception ros-
↪melodic-moveit-ros-planning-interface
```

(optional) We use catkin_tools. for building the package, you can install it as follows:

```
pip install --user catkin_tools
```

- Clone the repository in your ROS workspace:

```
cd catkin_ws/src
git clone https://github.com/francisc0garcia/braccio_camai
```

- Build the package:

```
catkin build --this
```

If the project build succesfully, we can continue with some tutorials for face and object detection and tracking.

If you have any problem, please check **Frequent questions** section for more information.

1.5 Tutorial: Face detection and tracking

Note: This tutorial assumes that the installation was completed successfully.

1.5.1 Components required for face detection and tracking

Face detection and tracking is composed by a sequence of tasks described as follows:

- Collect and publish images from USB webcam to ROS network.
- Detect faces on images using a face detection module based on Google coral Edge TPU. This module publishes a bounding box for each detected face.
- Track detected faces by generating arm commands, based on detections given as bounding boxes.
- Execute arm commands on Braccio Arm.

These tasks can be executed using the following ROS nodes:

- **robot_state_publisher**: Publish required transformations (TF) based on URDF model.
- **usb_cam**: Publish camera images to ROS network, using a USB webcam.
- **face_detection**: Subscribe for images and execute the face detection module using Google Coral Edge TPU.
- **DirectJointInterfaceBraccio**: Compute and publish arm commands, based on bounding box messages.
- **rosserial_braccio**: Connects to arduino using rosserial for executing commands on the Braccio arm.

1.5.2 How to execute face detection

A launch file containing required nodes is given on: **braccio_camai/launch/face_detection/braccio_driver_face_detection.launch**

It can be executed as follows:

```
cd catkin_ws
source devel/setup.bash
roscd braccio_camai
roslaunch  braccio_camai braccio_driver_face_detection.launch
```

1.5.3 How to test it

Once the launch file is executed, the Braccio arm should move to the initial position. If faces can be detected on images, bounding boxes are computed and corresponding arm commands are published. These commands are executed on the arm using the arduino interface.

A graphical interface based on RVIZ can be use for debugging and testing face detection and tracking. In a separate terminal, execute the following commands:

```
cd catkin_ws
source devel/setup.bash
roscd braccio_camai
roslaunch  braccio_camai braccio_gui_face_detection.launch
```

1.6 Tutorial: Object detection and tracking

Similar to face detection and tracking, different objects can be deteced and tracked using the Braccion arm. Supported object classes are taken from **mobilenet_ssd_v2_coco**.

1.6.1 How to execute object detection

A launch file containing required nodes is given on: **braccio_camai/launch/face_detection/braccio_driver_object_detection.launch**

It can be executed as follows:

```
cd catkin_ws
source devel/setup.bash
roscd braccio_camai
roslaunch  braccio_camai braccio_driver_object_detection.launch
```

Please notice that, object to be tracked can be parametrized on the ROS node **object_detection**, using the parameter **tracked_object**. Default object to be tracked is **bottle**.

1.6.2 How to test it

A graphical interface based on RVIZ can be use for debugging and testing object detection and tracking. In a separate terminal, execute the following commands:

```
cd catkin_ws
source devel/setup.bash
roscd braccio_camai
roslaunch  braccio_camai braccio_gui_object_detection.launch
```

1.7 Tutorial: Moveit

TODO

1.8 Frequent questions

This section presents some common problems and how to solve them!

1.8.1 Installation and compilation problems

- If you want to grant root access to GPIO for any executable:

```
cd devel/lib/robot_micro    # cd to the directory with your node
sudo chown root:root my_node     # change ownship to root
sudo chmod a+rx my_node          # set as executable by all
sudo chmod u+s my_node           # set the setuid bit
```

- If you want to remove password with sudo for current user:

edit: .. code-block:: none

sudo nano /etc/sudoers

add to end file: *user_name ALL=(ALL) NOPASSWD: ALL*

```
ubuntu ALL=(ALL) NOPASSWD: ALL
```

- If you want to grant root access to cameras and serial ports:

create a udev rule file:

```
sudo nano /etc/udev/rules.d/40-permissions.rules
```

insert content:

```
KERNEL=="ttyUSB0", MODE="0666"
KERNEL=="ttyUSB1", MODE="0666"
KERNEL=="ttyUSB2", MODE="0666"
KERNEL=="ttyUSB3", MODE="0666"
KERNEL=="ttyACM0", MODE="0666"
KERNEL=="video0", MODE="0666"
```

For Raspberry pi Cam:

```
KERNEL=="vchiq", MODE="0666"
```

1.9 Contact us

- If you want more info about the project, we are happy to contact you!

Francisco J. Garcia R. - francisc0garcia [at] outlook [dot] com

Developed by:

- Francisco J. Garcia R. 2019