3.1 Setup Guide

A list of the necessary parts for this setup.

- Ball Balancing System with PCB, Servos and Pixy2 camera
- 12 V power supply
- Micro-USB cable
- PC
- Checkerboard for camera calibration
- Ping Pong ball

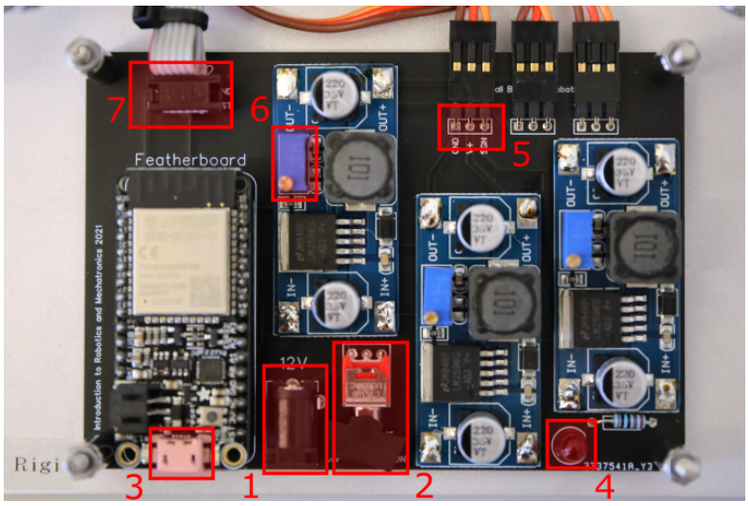


Figure 3.1: **PCB Overview.** 1) 12 V connector, 2) Servo power switch, 3) Featherboard Micro-USB connector, 4) Servo power LED, 5) Servo connector, 6) DC-DC voltage converter adjusting screw, 7) Pixy2 SPI connector

3.1.1 Software Setup

On the PC, four main software components are required

1. To have direct access to the serial port, we recommend using the serial monitor integrated in the Arduino IDE. The Arduino IDE can also be used to flash a different code onto the Featherboard, but this is *not recommended* since it could cause mechanical damage to the servo.

2. To calibrate the Pixy2 camera, the PixyMon Application has to be installed. Make sure *not* to install it on a virtual machine, but on the host, since virtual machines are not supported by PixyMon.

3. For the controller, several options are possible. The setup we use is a VMWare virtual machine which runs Ubuntu 18.04. We are using a controller in C language for educational reasons, but any software which allows serial communication could be used with the system.

4. We are using the Camera Calibrator application by Matlab. This only applies to the IRM lab course and is not a mandatory component. Make sure to have a working Matlab installation and install the *Computer Vision Toolbox* add-on.

Follow the documentation provided on the application's websites and install the necessary components. Their usage will be demonstrated and tested in the following paragraphs.

3.1.2 Servo Setup

1. Make sure the plate is mounted on the magnetic joints.

2. Connect the Featherboard on the BBS to the PC using the USB cable. If using a virtual machine, make sure it is detected correctly.

3. Connect the 12 V power supply to the PCB, turn on the switch on the PCB. The red LED on the PCB should light up.

4. In the Arduino IDE, open the Serial Monitor. Make sure to use the same settings as in Figure 3.3. The servo command syntax is

   ```
   C [degServoA] [degServoB] [degServoC]\n\r
   ```

   Once the command is entered, the servos should move to the specified angles.

5. When you enter `C 0 0 0`, the servos move to their zero position, where the first link should be horizontal. If you notice a large offset, unscrew the servo arm and attach it in a better position. Fine calibration should be done software-side in the controller, since the number of teeth of the servo connector limits the calibration accuracy.

3.1.3 Camera Setup

The correct camera setup is crucial! If the external lighting conditions change, the calibration should be repeated. Also if the controller does not perform as expected, or sudden jumps are observed, the camera setup should be checked first.

1. Mount the plate on the legs. Use the Arduino IDE as indicated in subsection 3.1.2 to move all three servos to a 30 degree position.

2. Unplug the Micro-USB cable from the Featherboard. Plug it into the Micro-USB connector of the Pixy2, which is located on the right side of the Pixy2.

3. Make sure you have the PixyMon software installed as indicated in subsection 3.1.1 and launch it. If the Pixy2 is connected correctly, you should see the camera image feed in the PixyMon GUI.

4. To teach an object to the Pixy2 camera, read through the documentation on the website. Additionally, tips on improving detection accuracy are provided. Refer to Figure 3.2 for an example set of settings.

5. Close PixyMon and unplug the Pixy2 camera. Connect the Micro-USB cable to the Featherboard again. Now the camera communicates with the Featherboard directly over its SPI port (see **??** and sends the coordinates to the Featherboard. To grab the coordinates, the character command `P` is used. Compare to Figure 3.3 to check whether you obtain useful results.

3.1.4 Matlab Camera Calibration

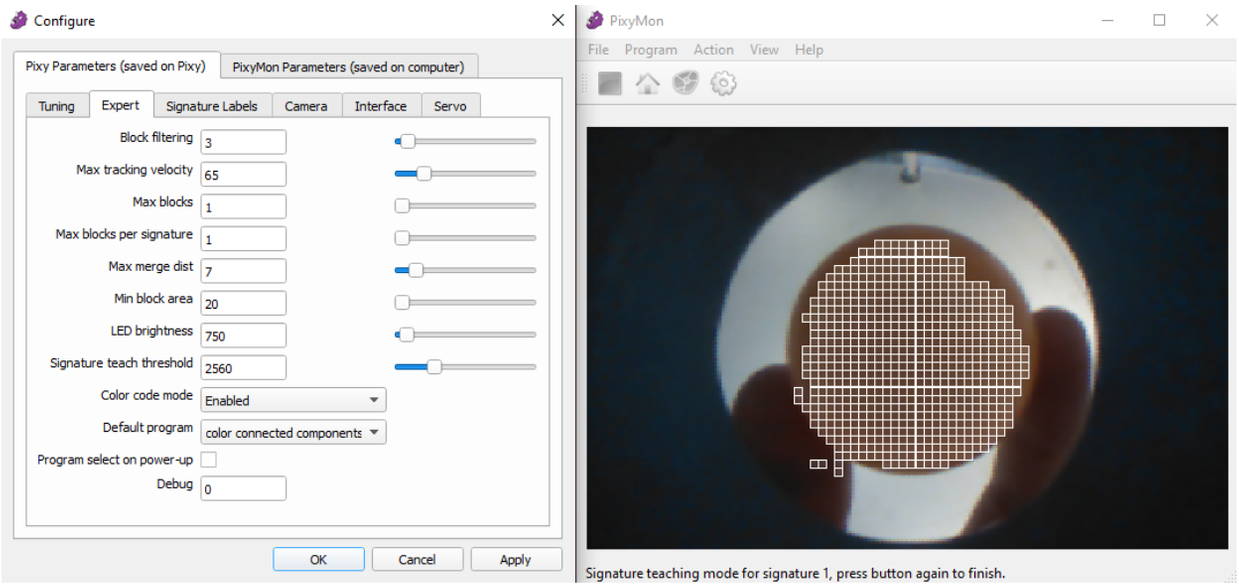
This part only applies if you are following the IRM lab course. To obtain the physical coordinates of the ball, the pixel coordinates need to be transformed. Matlab's Camera Calibrator App allows an automated calibration based on a set of sample images. After optimization, the intrinsic and extrinsic camera parameters can be saved to the workspace for further use. Here, only the calibration process to obtain is explained, the actual calibration equations are placed with the lab notes.

1. Print out the provided checkerboard and fix to a stable surface, for example a piece of cardboard.

2. Unmount the plate.

3. Start PixyMon. Before using the checkerboard, we have to adjust the PixyMon window size, otherwise the calibrator app in Matlab will not work. In the default window size, take a picture using File > Save Image. The image will be stored in your documents directory in a folder named PixyMon. Navigate to the folder, and inspect the image size of the catpured image. Now resize the window and repeat the procedure until the image size is about 1100×750 pixel.

a)



b)

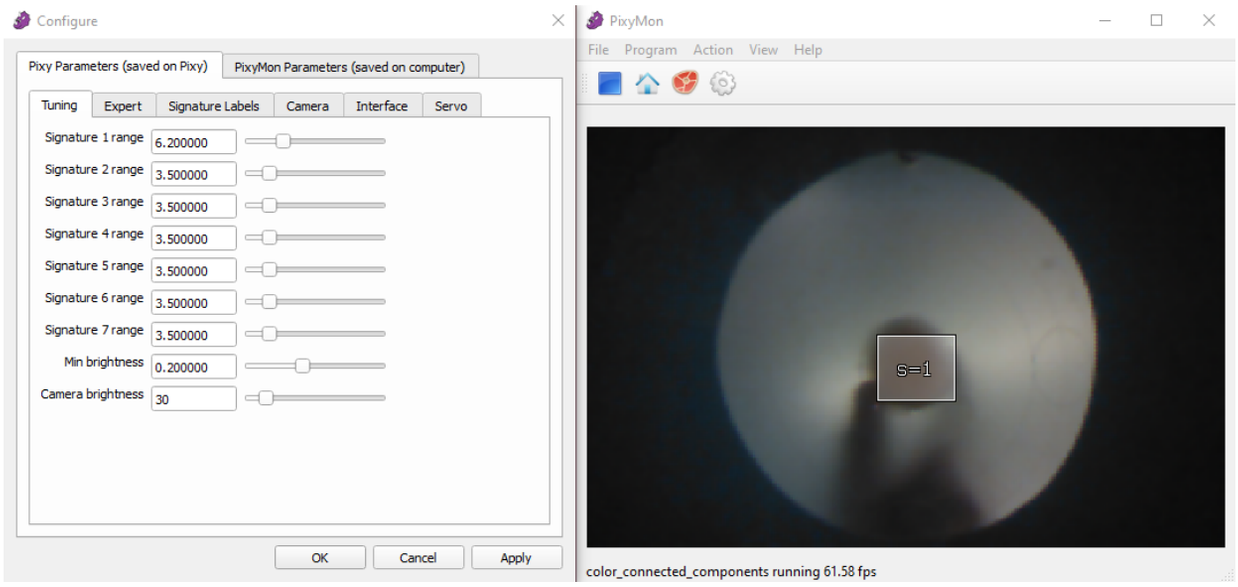


Figure 3.2: **Signature teaching with PixyMon.** a) Signature teaching mode. The Signature teach threshold is adjusted such that only the ball is detected. Also, there is only one block and one signature set to be detected. b) Once the signature has been set, the Signature 1 range is adjusted such that the ball is well detected, but no false positives appear.

4. Use File > Save Image to take about 30 images of the checkerboard, while slightly varying the checkerboard orientation and position. Make sure that the checkerboard is well visible on the picture, because the low resolution makes it difficult for the calibration algorithm to recognize the pattern.

5. Launch Matlab and make sure you followed subsection 3.1.1 to install the Computer Vision Toolbox. Open the Camera Calibrator app entering `cameraCalibrator` into the Matlab command line. Take a look at the application documentation to get familiar with the tool. Once the pictures are loaded, the calibration points should be marked as depicted in Figure 3.4.

6. Use the pictures you captured to calibrate the camera. Make sure to use the *Camera Model: Standard*. Once the calibration is done, you can export the camera parameters to the workspace and take a look at them. *Note:* You might notice that two parameters for the focal length f are available. If our camera had rectangular pixels, these values would account for this loss of symmetry. In our case, the pixels are square,

and therefore you can take the mean of the two focal lengths.

7. Remember that the coordinates returned by the camera do not necessarily correspond to the image size, since PixyMon saves its pictures depending on the window size. On the Pixy2 datasheet, you find the coordinate resolution, and the Matlab camera parameters contain the calibration image resolution. Make sure to scale the data accordingly when using the parameters for camera calibration in your code. *Note:* The imageSize vector in Matlab is defined as nrows $\times$ ncols, so the width of the image, which corresponds to the x-coordinate, is the second element of the vector. All other parameters are ordered as x, y.

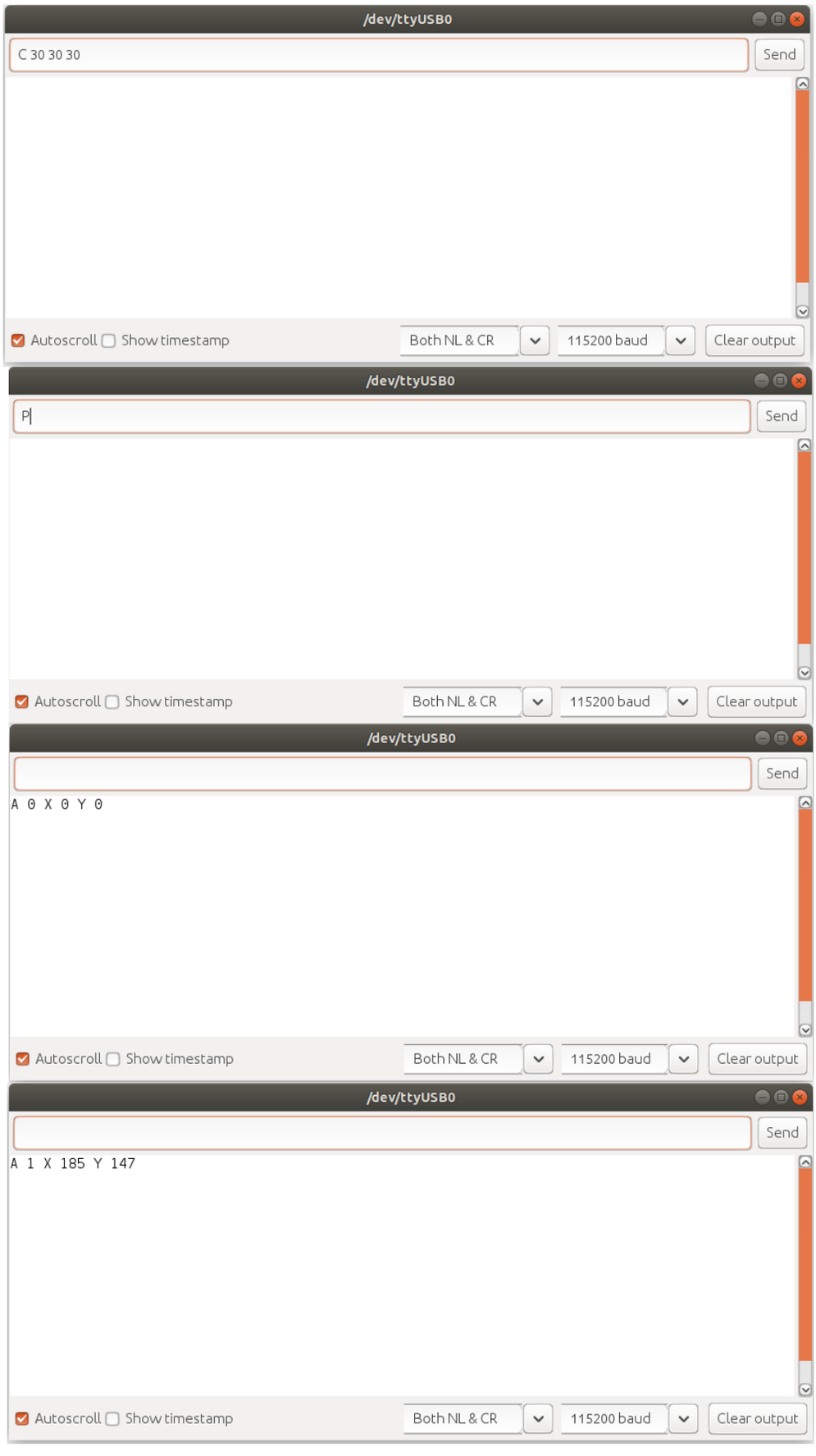


Figure 3.3: **Examples of serial input and output using the Arduino serial monitor.** Make sure to have the correct baudrate and line-terminatino settings. a) Servo command for 30 degree angle. b) To read the Pixy coordinates, enter the character `P`. c) If no object is detected, the first number after the `A` character is 0. d) If an object is detected, the first number is 1, followed by the X and Y coordinates in pixel units.

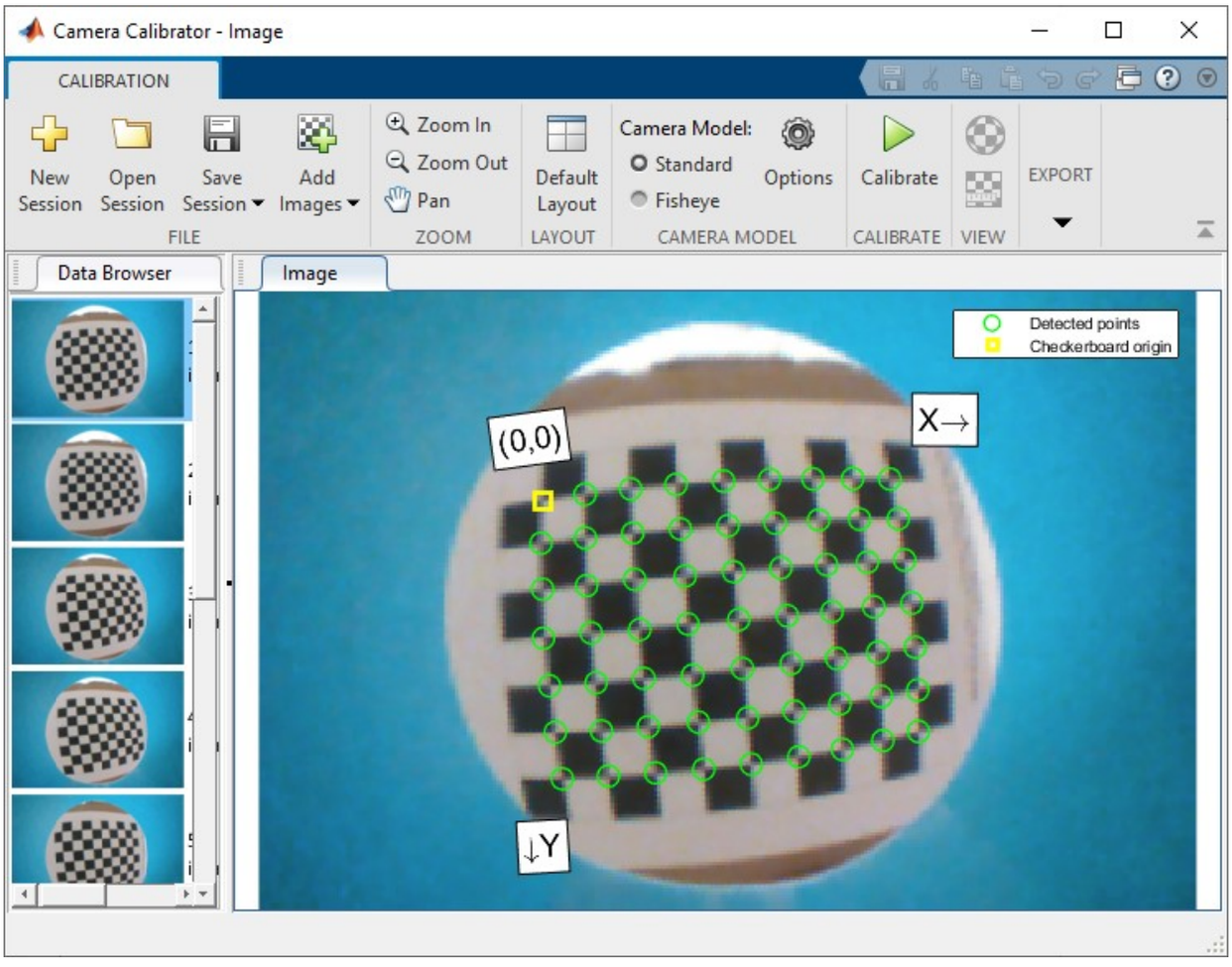


Figure 3.4: **Matlab Camera Calibrator App.** When the images are loaded, the edges of the checkerboard are marked with green circles. Once you clicked on the *Calibrate* button, the estimated edge locations will appear in red and the calibration parameters are available.

Figure 3.5: **Image capture with PixyMon.** Take images for camera calibration. The checkerboard should be well visible and PixyMon should be executed in full-screen mode.