Robotic Arm Control Through Serial and WIFI Communication

Standard Operating Procedure

Jonathan Dean and John Zoodsma

Table of Contents

[II. Project Purpose 4](#_Toc529813833)

[III. Setup 4](#_Toc529813834)

[A. Hardware 4](#_Toc529813835)

[1. Mechanical 4](#_Toc529813836)

[2. Electrical 4](#_Toc529813837)

[B. Software Setup 5](#_Toc529813838)

[1. Arduino Programming 5](#_Toc529813839)

[2. Raspberry Pi Server 5](#_Toc529813840)

[3. Raspberry Pi Client (Remote) 5](#_Toc529813841)

[4. Raspberry Pi Client (Serial) 6](#_Toc529813842)

[IV. Usage Instructions 6](#_Toc529813843)

[A. Controlling the Arm 6](#_Toc529813844)

[1. Generating Commands 6](#_Toc529813845)

[2. Sending Commands 7](#_Toc529813846)

[V. Description 7](#_Toc529813847)

[A. System Description 7](#_Toc529813848)

[1. General Description 7](#_Toc529813849)

[B. Hardware Diagram 8](#_Toc529813850)

[C. Communication Diagram 8](#_Toc529813851)

[VI. Classes 8](#_Toc529813852)

[A. Purpose 8](#_Toc529813853)

[1. Versatility 8](#_Toc529813854)

[2. Ease of use 8](#_Toc529813855)

[3. Standardization 9](#_Toc529813856)

[B. PI\_Srvr - Server Class 9](#_Toc529813857)

[1. Class Description 9](#_Toc529813858)

[2. Function Descriptions 9](#_Toc529813859)

[C. PI\_Cli - Client Class (Remote) 9](#_Toc529813860)

[1. Class Description 9](#_Toc529813861)

[2. Function Descriptions 9](#_Toc529813862)

[D. PI\_SerCli - Client Class (Serial) 10](#_Toc529813863)

[1. Class Description 10](#_Toc529813864)

[2. Function Descriptions 10](#_Toc529813865)

[VII. Version 10](#_Toc529813866)

# Project Purpose

The purpose of this program and device set is to control a robotic arm over WIFI from a remote location. In order to do this, we are sending commands over WIFI that will be transmitted to the robotic arm using serial communication to control the robotic arm.

# Setup

## Hardware

### Mechanical

#### The Arduino and the Raspberry Pi (intended for serial communication with the Arduino) should be mounted on the same surface as the robotic arm.

### Electrical

#### Connect the Arduino to the robotic arm using the mounted motor driver shield.

#### Power the Arduino using a 7-12V DC power supply.

#### Power two Raspberry Pi’s using 5V DC power supplies.

##### A Raspberry Pi cannot supply enough power to the Arduino to be used as a power source for it, however we can program the Arduino using the Raspberry Pi over USB.

##### One Raspberry Pi will be used for serial communication (Serial) with the Arduino, and the other Raspberry Pi will be used for machine learning (Remote).

##### Both Raspberry Pi’s will communicate to each other and a server over WIFI

#### Plug the Arduino into the Raspberry Pi (Serial) using USB.

##### **Begin Software Setup Instructions (Step B) at this point**

#### Connect the mounted motor driver shield to a 12V DC Power supply to power the robotic arm.

#### Make sure that the push button switch (blue) on the motor driver shield is pushed in.

##### The arm should be powered and ready to receive commands at this point.

## Software Setup

### Arduino Programming

#### Open the Arduino IDE.

#### Open file “controller\_1.ino”.

#### In the IDE, set the target board to be the Arduino Mega 2560.

#### Verify and program the device.

#### The Arduino should now be ready to receive commands serially.

### Raspberry Pi Server

#### The server can be hosted on either raspberry pi or on a different device entirely.

#### Open a new terminal instance and navigate to the file location of the file “PI\_Srvr.py”.

#### Open Python in the terminal.

#### Import the PI\_Srvr (server) class.

##### from PI\_Srvr import \*

#### Create a server variable with a port number passed as a parameter.

##### server = PI\_Srvr(10001)

###### This opens a server on port 10001

### Raspberry Pi Client (Remote)

#### Open a new terminal instance and navigate to the file location of the file “PI\_Cli.py”

#### Open Python in the terminal.

#### Import the PI\_Cli (client) class.

##### from PI\_Cli import \*

#### Create a client variable with an ip and a port number passed as parameters.

##### client = PI\_Cli(“127.0.0.1”, 10001)

###### This will attempt to connect to a server at IP address 127.0.0.1 (localhost) on port 10001.

#### The client should be connected to a server now.

### Raspberry Pi Client (Serial)

#### Open a new terminal instance and navigate to the file location of the file “PI\_SerCli.py”

#### Open Python in the terminal.

#### Import the PI\_SerCli (serial client) class.

##### from PI\_SerCli import \*

#### Create a serial client variable with an ip, a port number, and a baud rate passed as parameters.

##### ser\_client = PI\_Cli(“127.0.0.1”, 10001, 115200)

###### This will attempt to connect to a server at IP address 127.0.0.1 (localhost) on port 10001 and connect serially to the Arduino communicating at a baud rate of 115200.

#### The client should be connected to a server and to the Arduino now.

# Usage Instructions

## Controlling the Arm

### Generating Commands

#### The robotic arm can be sent commands on where to move the robotic arm specified by degree positions.

#### Each command specifies a position and a servo to move.

#### Servos are given values ‘a’ through ‘f’ labelled from bottom to top servo.

#### To move the robot, a command has the format “<degrees><servo\_label>”

##### Ex. A command that would move servo A to position 85° would be “85a”

#### Multiple commands can be sent using commas and spaces.

#### Ex. “45a, 85b, 25c, 89d” etc.

### Sending Commands

#### Commands can be sent from a remote client.

#### If a remote client is connected, then commands can be sent as a string using the PI\_Cli class’s Send\_Msg function. Examples shown below.

##### client.Send\_Msg(“Hello”)

###### This would send a message saying hello to any clients listening.

##### Client.Send\_Msg(“85a, 35b, 120c”)

###### This would send a message that the serial client could interpret.

###### The message translates as move servo A to position 85°, move servo B to position 35°, and move servo C to position 120°.

# Description

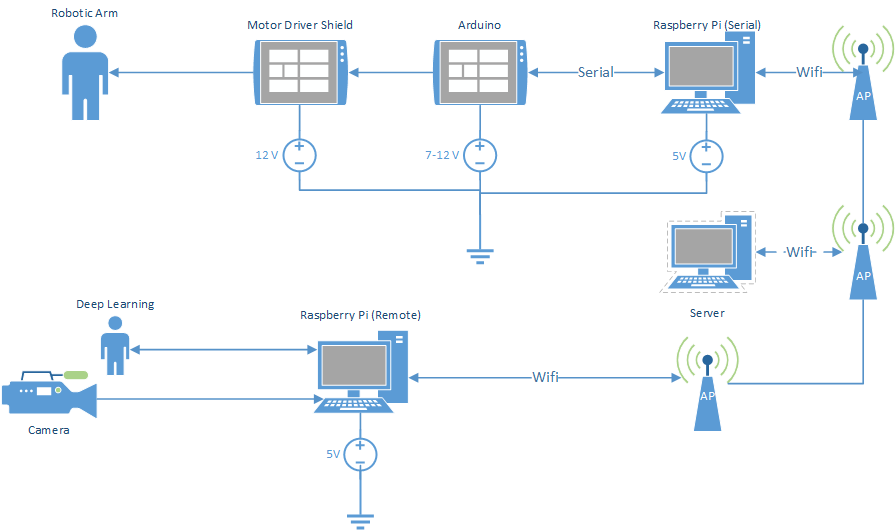
## System Description

### General Description

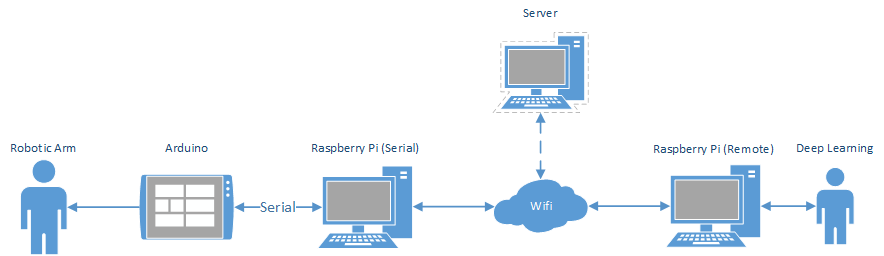
#### A Raspberry Pi is used remotely to send commands to another Raspberry Pi. The second Raspberry Pi will relay the commands to an Arduino using serial communication. The Arduino has a program that will allow it to process the commands and to control the arm’s movement.

#### Deep learning is used for sending commands and decision making for our design. The decision making process for moving the robotic arm are not currently done locally or from the server.

## Hardware Diagram



## Communication Diagram



# Classes

## Purpose

### Versatility

#### The server and client functions are implemented using classes to allow for use in other programs, rather than being purely dedicated to being a server or a client.

### Ease of use

#### The server and client programs are easy to embed into a program if desired.

#### Development using these server/client classes allows for standardization between different programs.

### Standardization

#### Using classes that are identical in structure will allow us to develop simultaniously with the deep learning team and reduces the chance of error.

## PI\_Srvr - Server Class

### Class Description

#### The PI\_SRVR class describes a Multi-Client server and its Functions.

### Function Descriptions

#### \_init\_

##### Inputs : port number (integer)

##### Outputs: Server Variable (PI\_Srvr)

##### Ex. server = PI\_Srvr(port\_num)

## PI\_Cli - Client Class (Remote)

### Class Description

#### The PI\_Cli class describes a wifi communication Client and its Functions.

### Function Descriptions

#### \_INIT\_

##### Inputs : IP address (string), port number (integer)

##### Outputs: Client Variable (PI\_Cli)

##### Ex. client = PI\_Cli(ip\_addr, port\_num)

#### Send\_Msg

##### Inputs : message (string)

##### Outputs: N/A

##### Ex. client.Send\_Msg(msg)

## PI\_SerCli - Client Class (Serial)

### Class Description

#### The PI\_SerCli class describes a Serial and WiFi communication client and its Functions.

### Function Descriptions

#### \_INIT\_

##### Inputs : IP address (string), port number (integer)

##### Outputs: Client Variable (PI\_SerCli)

##### Ex. serial\_client = PI\_SerCli(ip\_addr, port\_num)

#### Send\_Msg

##### Inputs : message (string)

##### Outputs: N/A

##### Ex. serial\_client.Send\_Msg(msg)

#### serial\_write

##### Inputs : message (string)

##### Outputs: N/A

##### Ex. serial\_client.serial\_write(msg)

# Version

**Version : 1.0.0**

**Date: 11/12/2018**