Remote as a Service

With a small payment of only 10$ per use you too can pay us to install software on your Arduino and Raspberry Pi.

```
#define Raspberry-Pi Master
#define Arduino Slave
```

Requirements

> Make use of two Arduino (or similar) micro-controllers

We have one Raspberry Pi and one Arduino. The Arduino will communicate with the Raspberry Pi over serial, and act as a Remote Control over Serial Communication. The Arduino will host a web-server with a REST Api that will allow control over the Arduino controller that it will power.

> Utilize at least four different external devices (16x2 display, LEDs, pushbuttons, touch sensors, etc)

- LED on the Arduino
- IR Emitter on the Arduino
- Computer that's running Python Script with Alfred.
- TV that is handling input/output

If those are not good enough:

- Phone running iOS App to control Arduino
- Webpage running Website to control Arduino

If those are not good enough:

- 16x2 Display to output signal name on Slave
- Pushbutton to emit some signal on Slave

> Utilize at least one communication mechanism (ethernet, bluetooth, serial, etc)

We have a Serial Communication between the Raspberry Pi and the Arduino, a Ethernet/Wi-Fi Communication between the Raspberry Pi and a python script on the computer that is hooked up to Alfred 3 (Can you tell why I picked this project?) and a IR communication from the Arduino to the TV/Other.

> Involve some original work

We hope so!

Details

Master -> Slave Serial Interface

Since the Slave can't get success/failure or need to send any data back, we can have a one way communication to the Slave.

Command will be sent from the Master to the Slave using Serial Commands. The Arduino will listen for commands, and if any are received will immediately execute the corresponding command.

Web-Server

A web-server running on the Master will let us control the remote easily from a variety of devices. Making HTTP Requests is easily handled, and even more so if I can get Bonjour to easily detect Remotes on the same Wi-Fi Network (Zero-Conf Detection).

This Web-Server will be written in:

- Go? (Never used it and would like to try)
- Rust? (Never used it and would like to try)
- Kotlin? (Used it a bunch, know it has WebServer libraries) (Might be too slow (JVM))
- Swift? (Linux Support isn't great)
- Node.js? (Might be too slow)

Optional:

Write a web-ui for the Web-Server, allowing for the remote to be controlled without any clients installed on a device.

Clients

Clients will be easy to write:

- Python Script that does simple HTTP Requests
- Alfred 3 Workflow to control the Python Script
- iOS App that just sends HTTP Requests

- Mac Menubar App that does simple HTTP Requests
- Windows Menubar App that does simple HTTP Reqests

Since HTTP Requests are dead simple in basically every language and these are one-way controls it should be easy to write clients in whatever we want.

Timeline

Step 1: Arduino IR Emitter

If we can get this step done, then the biggest unknown of the project is over. Once we can get the IR Emitter to turn on/off a TV and/or control the volume/input of a TV, then we can start to control it.

Step 2: Raspberry Pi -> Arduino

At this point, we can work on creating the link from the Arduino to the Raspberry Pi. If this works, then we can move all development to the more capable Raspberry Pi and leave the Arduino as a basic accessory to the Raspberry Pi

Step 3: Raspberry Pi Webserver

Once the Pi can control the Arduino, we can then move onto setting up the REST Api for controlling the Remote.

Step 4: Clients

However many of these we need to fulfill requirements.

Tutorials

Basic tutorial for using the IR Led / IR receiver

More in depth with Library Included

Kotlin -> Raspberry Pi

We have confirmed that we are using Kotlin, since Kotlin/Native looks interesting.

Here are some of the tutorials we are going to be using to get this working:

Kotlin with GPIO

We are going to be using WiringPi to intern with the pins on the Raspberry Pi Side.

Wiring Pi

This will let us handle the Serial Port with little/no problem. That was one of the biggest problems between the raspberry pi -> Arduino -> Kotlin communication.

Kotlin -> Serial -> Arduino

The Arduino isn't powerful enough to run Kotlin. This restricts us from being able to run the same code on both devices. However, with the more complicated code running on the Raspberry Pi this isn't much of a problem, since we need a nicer program there than on the Arduino.

Kotlin -> HTTP Server

We have an example of the HTTP Server in Kotlin/Native that we can use for the entire project. dhere.