

**Project System Specification**

Gabriel Almeida

Ashraf Drieas

Juan Inzunza

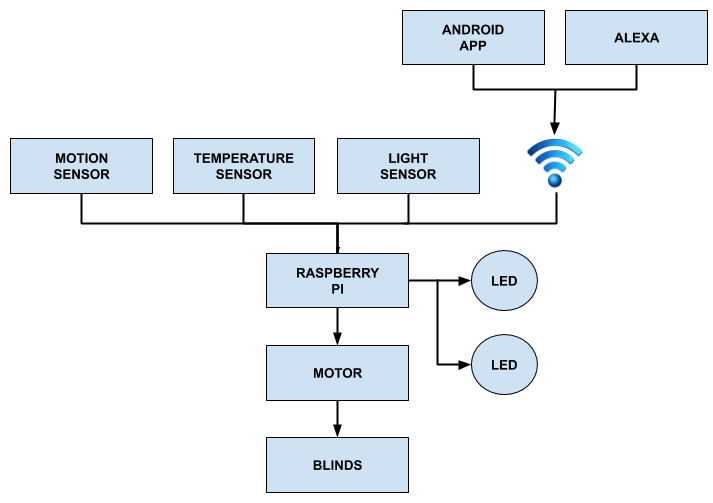
Brandon Fins

Mandy Ledford

**Concept:**

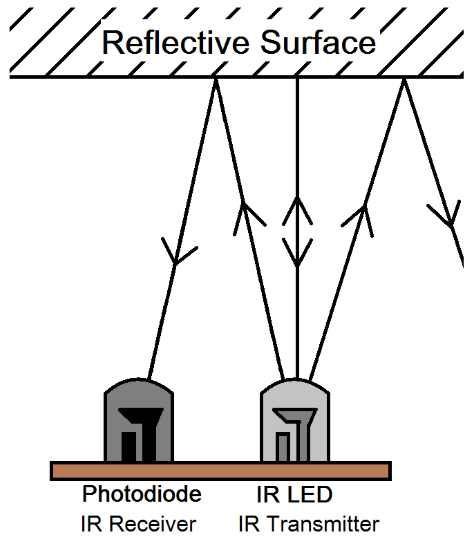
The Easy-Blinds system block diagram is outlined below. The Easy-Blinds system contains a fully automated and controllable system. Our device includes the following components: a standard set of faux wood blinds, a servo used for opening and closing the blinds, a Raspberry Pi, three sensors (an infrared motion sensor, a photo-resistor that serves as a light sensor, and a temperature sensor), a downloadable EasyBlinds App to interface with the blinds, and a voice communication module. All physical components of the Easy-Blinds device will be installed in the top compartment of the system. When the blinds are open, the user can wave their hand near this sensor and the blinds will automatically close. The light sensor will detect ambient light entering the immediate area, and based on sensitivity setting, will automatically open or close the blinds. Similarly, the temperature sensor will detect ambient temperature and based on sensitivity settings, adjust the blinds accordingly. The blinds also feature two external means of control, the voice communication module and the Easy-Blinds App. The voice communication module allows for the blinds to be opened, closed, and set into automatic mode through the use of simple voice commands. The Easy-Blinds App allows the user to open and close the blinds, set the blinds in automatic or manual control, and set the light and temperature sensitivity of the blinds. With these features, the user can toggle the Blinds from automatic to manual control, and the user can open and close the blinds at will.

**Block Diagram:**



**Block Diagram Description:**

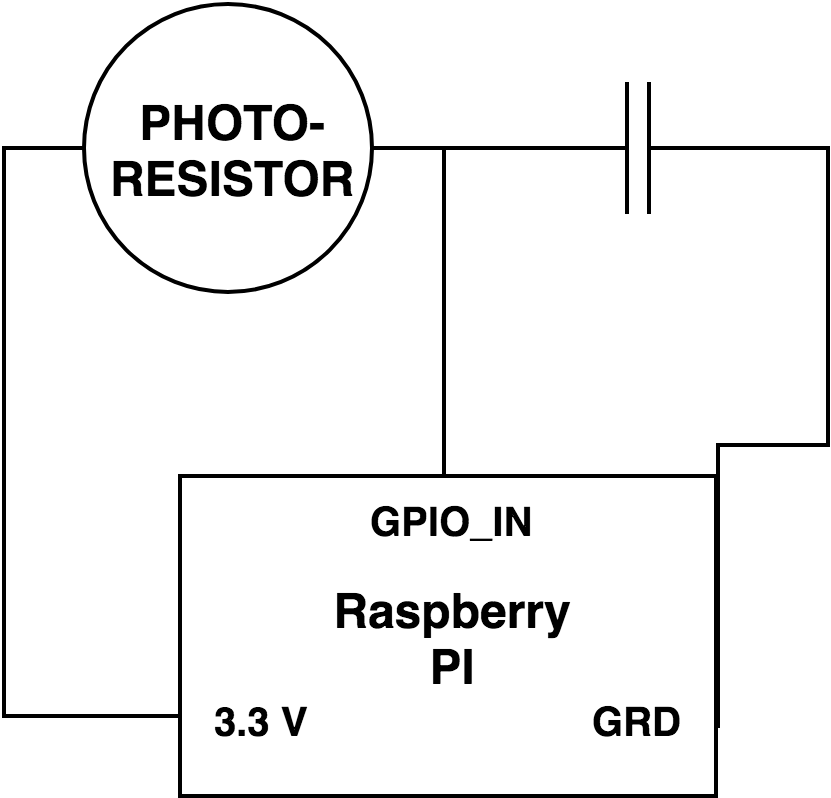
1. **Motion Sensor**

The motion sensor we are using for our system is the Infrared Barrier Sensor Module model SEE00306I.

This sensor can detect objects at a distance of 2-30 cm, which can be adjusted using the potentiometer built into the sensor. It can detect objects at an angle of up to 35 degrees and has a working voltage range of 3.3v-5v. The bourde size of the sensor is 3.2cm x 1.4 cm.

Ideally, our system will allow the user to activate or deactivate the motion sensing control. The motion sensor will be installed in the top compartment of the Easy-Blinds system. When manual control is active, the sensor will send an infrared signal that detects any type of motion within a range of 2-30cm. The user may open and close the blinds as many times as needed using the motion sensor while the system is in manual mode. When motion is detected, the sensor sends a low signal to the Raspberry pi to indicate motion is present and, depending on current state, will toggle open or close the blinds.

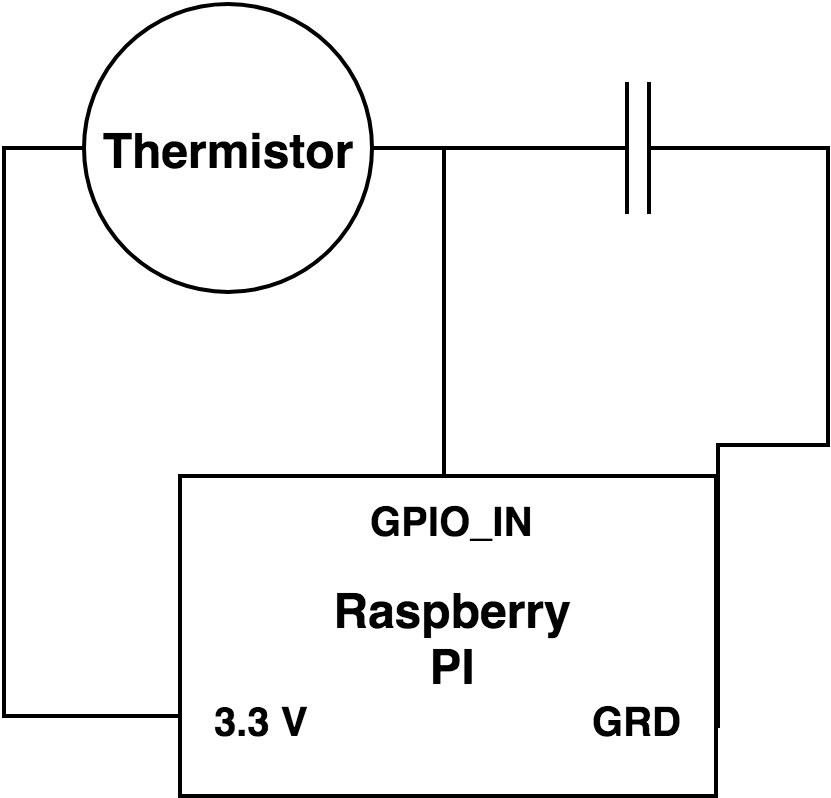
As mentioned before, we would like the user to have the option to turn on or off the motion sensing capabilities of the device. If the user is content having the blinds in the current state (open or closed) and they want to avoid any unintentional activation of the motion sensor trigger, they should be able to turn it off with the click of a button. This will be accomplished using the IR remote and receiver. The remote and receiver will allow the user to change the blinds in and out of auto and manual control.

1. **Light Sensor**

The sensor chosen for this project is a CdS photoresistor, 5-10k ohms of resistance when lit up, and near 200k ohms when dark.

A photoresistor is a light-controlled variable resistor. Which means that it changes resistance with a change in light intensity. Specifically, the resistance decreases as light intensity increases, and the resistance increases as light intensity decreases.

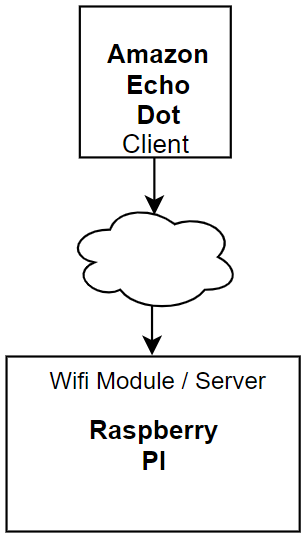
The raspberry pi GPIO inputs can only read signals as either high or low, so we can not directly read the output of the photoresistor. This is why the 1 uF capacitor is used in the light sensor module. The capacitor combined with a simple python script allows us to detect when the capacitor is at ¾ capacity and output a value associated with how long it took the capacitor to reach ¾ capacity. This allows us to detect variations in light intensity, because the resistance of the photoresistor will increase and decrease with changes in light intensity. Through testing we can find which values are output when the sun rises and which values are output when the sun sets. These values are then used to trigger the blinds to open and closed based on the current light intensity.

1. **Temperature Sensor**

The thermistor used in this project is the TDC 310, which has a resistance of 10k ohms at 25℃.

A thermistor is a temperature controlled resistor. Which mean that it changes resistance with a change in temperature. This is similar to the photoresistor described in the previous section, only instead of changing with light intensity it changes with temperature.

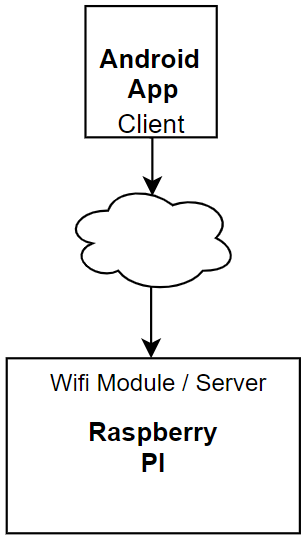
The circuit for the Temperature sensor is identical to that of the light sensor, except the Photoresistor is replaced with a thermistor. Therefore, the ¾ capacity of a 1 uF capacitor is used to detect variation in temperature. This is unfortunately not the most accurate way to measure temperature but for proof of concept it has worked well.

1. **Alexa communication**

For this project we’ll be using an Echo Dot for voice commands. It’s powered by an AC-to-DC 5V power adapter. The Echo dot is an Amazon smart device. As such it utilizes, Alexa, Amazon’s version of a personal hands free assistant.

The Amazon Echo Dot and the Raspberry Pi connection is established through WiFi. Communication between the two devices is established with the use of Fauxmo, a Python module. Fauxmo works by emulating Belkin Wemo devices, which the Echo Dot can either turn off or on, provided their on the same network. After which these “devices” can be found and registered. In our case, the Echo Dot registers each GPIO pin on the Raspberry Pi, as a device which it can control.

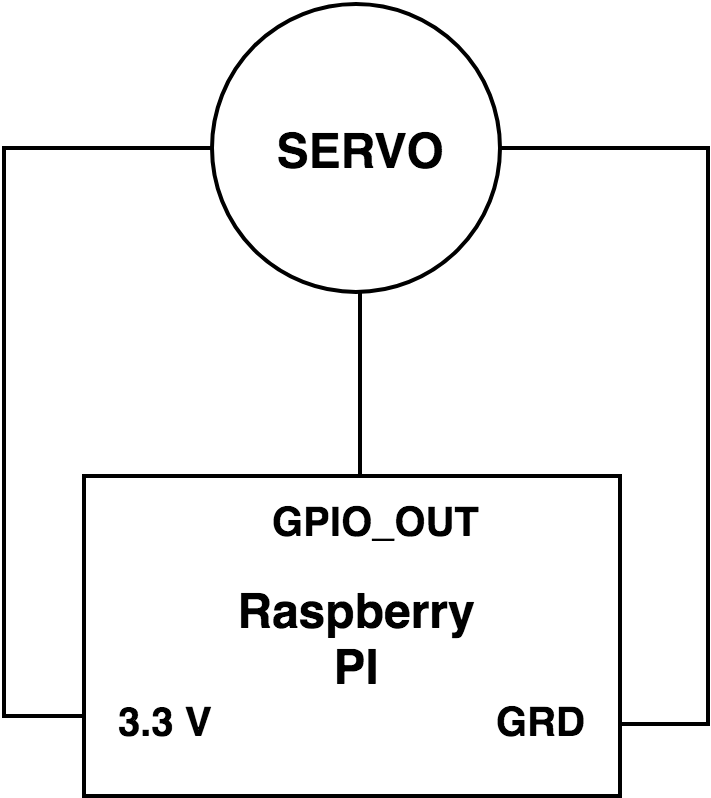
The Echo Dot will be implemented to use 2 “devices” or GPIO pins. These two pins are the pin for the motor and an unused “dummy” pin for the states/modes. The motor will be activated to open the blinds, and this can be done by saying “Alexa, turn on blinds”. Whereas to close the blinds the motor will be deactivated by saying the opposite “Alexa, turn off blinds”. The “dummy” pin mentioned earlier will be used to change the states/modes of the blinds. This pin will be unoccupied and no wires will be used or involved with it. Since the Echo Dot can only be used to turn on and off the GPIOs, this “dummy” pin will serve as the bridge between the Echo Dot and the current state/mode of the blinds. To put the blinds in auto mode, the command “Alexa, turn on auto mode” will be used. Then to put the blinds out of auto mode and into manual mode, the command “Alexa, turn off auto mode”. Both of these commands will be done by reading the high and low output of the “dummy” pin. When using Fauxmo, the “on” keyword corresponds to a high output, whereas the “off” keyword corresponds to a low output on the GPIO. In the case of the states/modes, the output will be read and if it’s being read as high on the “dummy” pin, that corresponds to changing into auto mode. Finally if the output being read on the “dummy” pin is low, this results in the changing into manual mode.



1. **Android App communication**

The Android App is build using Android Studio, and written in Javascript. It creates a serial port over wifi to interface with the Raspberry pi. The Raspberry pi acts as the server and checks for incoming data from the Easy-Blinds App which acts as the Client.

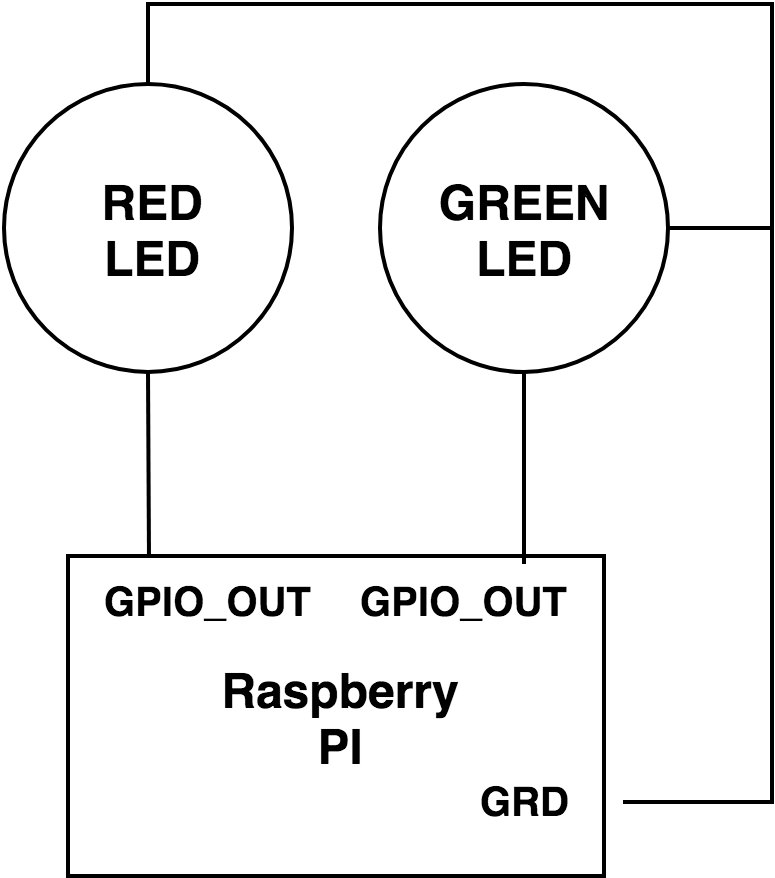
The App has six controls. Four are buttons and two sliders. The four buttons control, opening the blinds, closing the blinds, setting the system in automatic mode, and setting the system in manual mode. The two sliders control the values of the light intensity and temperature at which the blinds open and close when in automatic mode.



1. **Motor**

The motor we’re using in the system is the S3003 Servo Motor by Futaba. This motor uses three connection ports: power, ground, and input/output signal. We supply power to the motor from the 3.3V pin on the raspberry pi. Our S3003 has a turn speed of .23 sec/60° at 4.8V and torque of 44oz/in or 3.2kg/cm at 4.8V.

In our system, apart from the raspberry pi, the servo motor is the most vital piece. It is responsible for opening and closing the blinds according to the readings from the various sensors and wifi communication. The raspberry pi takes measurements in real time and when a certain reading from one our sensors goes past a set threshold, the motor responds accordingly, or if the pi receives a command from a client across wifi. It’s designed to open and close the blinds 90 degrees in either direction, based on what values are currently being read by the sensor or sent from the client.



1. **LED’s**

The two LED blocks in our block diagram are used as indicators to let the user know if the system is in Auto or Manual mode. When the System is in Auto mode a red LED is activated and when the system is in Manual mode a green LED is activated.

**System Analysis:**

We ran into a couple of problems while implementing the communications portion of the system design. The first issue we ran into was controlling the blinds through voice communication with the Echo Dot. The implementation of the interface we used, send data from the Dot and controls the GPIO pins. This only allowed us the ability to turn a GPIO on and off, but for the servo we need to control which way the servo moves. A simple turning on and off of a GPIO pin does not work for our system. The second issue we ran into when deciding which approach to take when setting up communication with the Android App. There are a lot of different methods to interfacing with the the raspberry pi, and we had never designed a system like this so it took a lot of research to settle on an approach.

**Project Plan:**