Annoying Alarm Clock

The alarm clock that will make you hate alarms even more.

Description:

As annoying as alarms are, we really need them. However, you may sometimes (or all the time) sleep through alarms. This can make you late for anything – interviews, meetings, school, when to take your pills, your wedding, etc. And we all hate to be late for everything. This can happen because we can never notice our alarms until it is too late.

This Annoying Alarm Clock will flash bright LEDs and play some loud audio to make sure you want to turn it off immediately. This can make sure you're alert, and notified that you need to go pick up your kids from school before forgetting about them again.

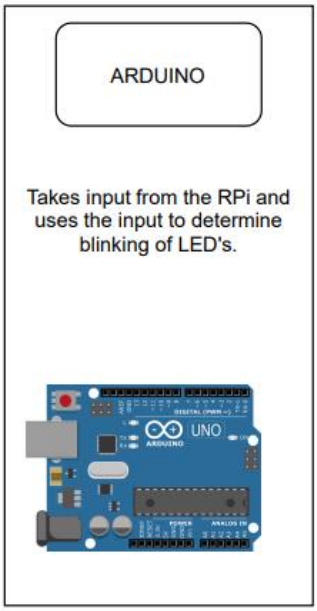
Purpose?

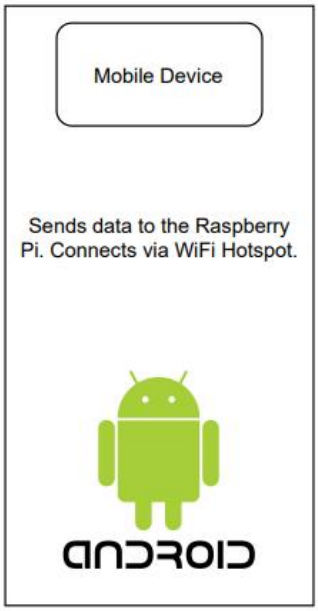
This project was more a learning experience. I've recently gotten an Arduino kit, and wanted to learn on how everything worked. I've also used Node Js for some things, but that will be more in detail later; and I have not used Node Js very often previously, so this was a new learning experience for me as well.

How Does It Work?

Brief Description:

In short, the Raspberry Pi hosts an HTTP server to take requests from a phone (which will be using the website hosted by the Pi). It can then take the data and check if it needs to turn on the alarm stuff, turn it off, or return some data back to the phone. The Arduino takes data from the Raspberry Pi, and determines if it needs to start blinking the LEDs, or if it needs to turn it off. The phone will simply allow the user to set an alarm for a time, and will communicate with the Raspberry Pi.





Mobile Phone

Tasks

- Allow the user to
  - Create a new alarm
  - Stop the current alarm
  - Cancel an alarm
- Send data over to the Raspberry Pi depending on what the user needs

Flow

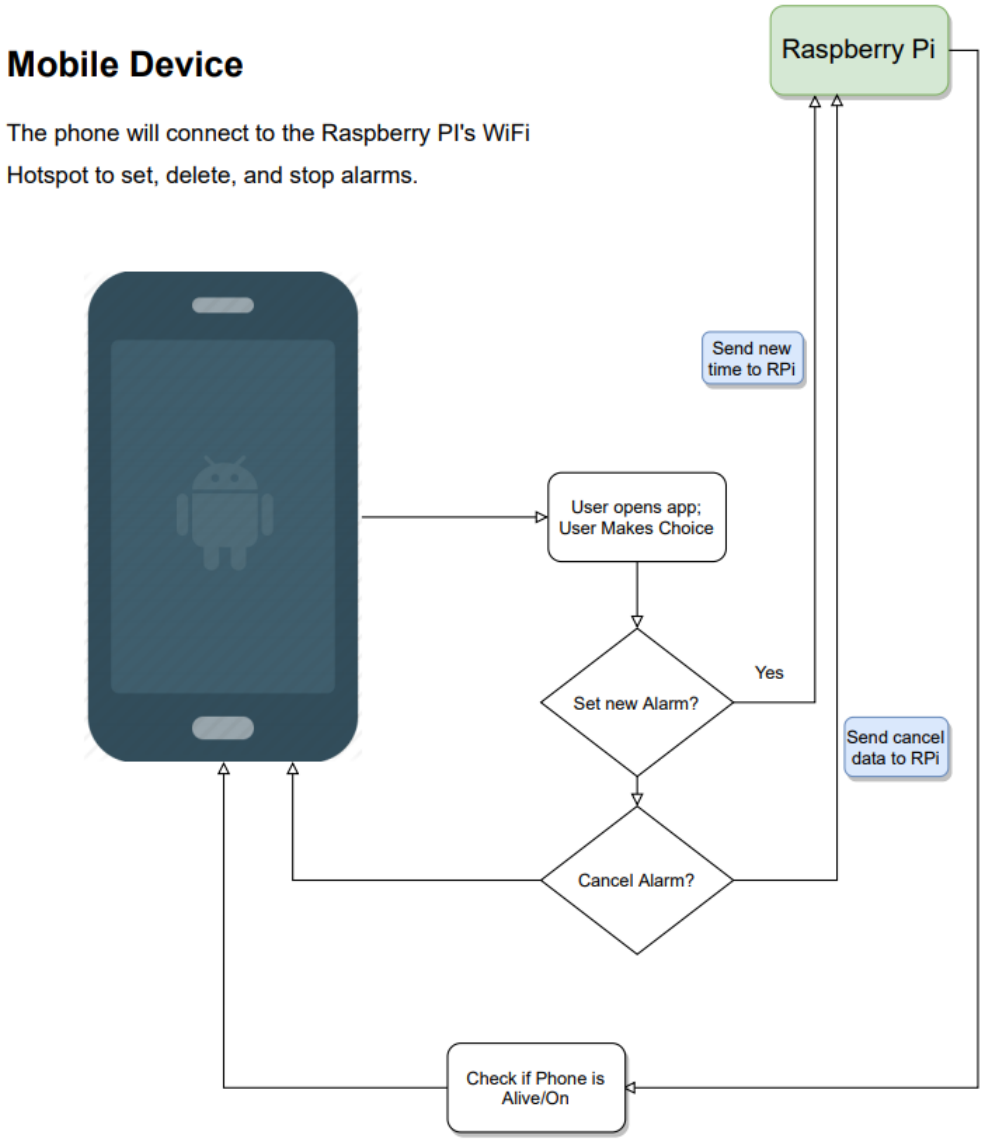
The phone will start up with a simple UI. Show the user the current time, and 3 buttons – one that allows the user to create a new alarm, another to cancel any alarms, and one other button that will allow the user to stop the alarm. There is no snooze button (sorry :/).

When the user taps on “Create Alarm”, it will prompt the user to enter what time they would like to set the alarm for. They have the option to cancel or set the alarm. If they decide to set the alarm, the app will send data over to the server to let it know that we are going to wait for a certain time to start the alarm.

If the user decides to cancel an alarm, the app will show a list of alarms the user may have set, and the user can disable all of them or specific ones.

When the alarm is going off, the user can tap the “STOP” button to stop the alarm.

Flowchart (Visual)



Raspberry Pi

Tasks

- Take requests from the phone
    - Add a new alarm
    - Stop the current alarm
    - Remove an alarm
- Communicate with the Arduino
    - Check its status if it
        - Is currently flashing lights
        - Is connected to the raspberry pi
    - Tell it to turn on/off the LEDs
- Check the timer(s) if they are ringing or not (yet)

Flow

The Raspberry Pi hosts its own WiFi network. This would mean the phone is required to connect to the Raspberry Pi's WiFi in order for it to work. The server will listen for incoming data from the phone. Once the Raspberry Pi receives a POST request from the phone, the Raspberry Pi will manage that data.

The server has 2 main scripts running. One being the main Web Server, and the other being a python script that checks if one of the times matches the current time.

If the phone wants to add a new alarm, the Raspberry Pi will append to the list of alarms (if there are multiple active ones). In the background, the Pi will check it's time. If the time has reached an alarm, the Pi will attempt to see if the time is correct as well on the Phone. If the time matches, the Pi will send a message to the Arduino of "RedE". The Pi will wait for a response back from the Arduino. If the Pi receives the message "Ready", the Pi will send another message to initialize the lights, "STARTL" (I think it's funny

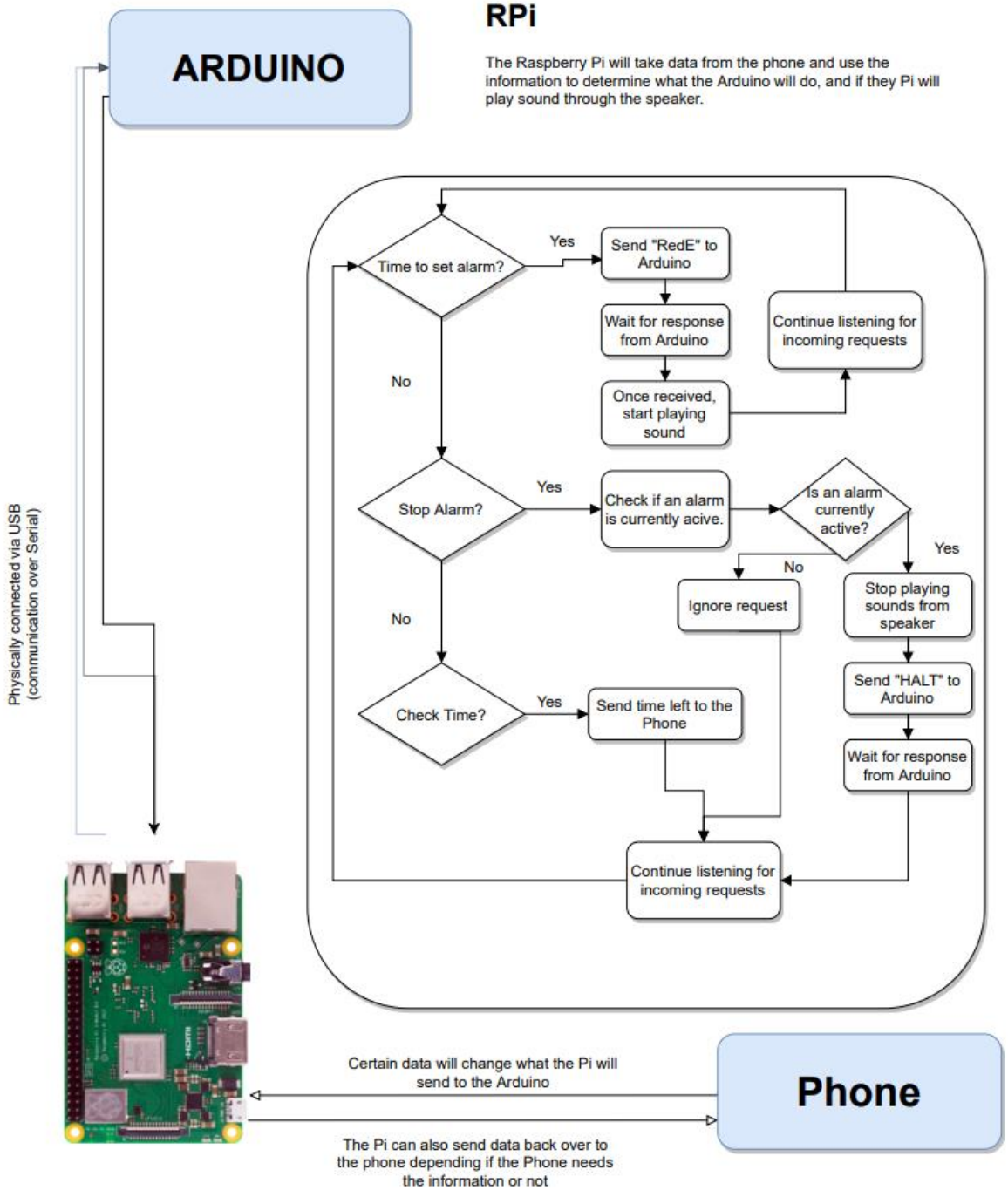
that it's said like "startle", although it's supposed to mean "Start LEDs" and it is an alarm, so it'll startle you). Then the Pi will wait for a message back saying "StartingL", then the pi will start a background script in Python to play noises. This is so that the Pi can still receive incoming data from the Phone to turn it off.

If the phone wants to stop the alarm currently going off, the Pi will check with the Arduino if it is currently flashing, "CHECKL".

If the Pi receives "STATUS:LIVE", the Pi will run a script to turn off the LEDs via "HALT", and stop the script that is running the sounds. The Pi will then send back to the phone that the alarm is stopped. Else if it receives "STATUS:OFF", the Pi will ignore and continue with what it is doing.

If the phone wants to disable/remove an alarm, the Pi will remove/disable an alarm in a list of alarms.

Flowchart (Visual)



Arduino

Tasks

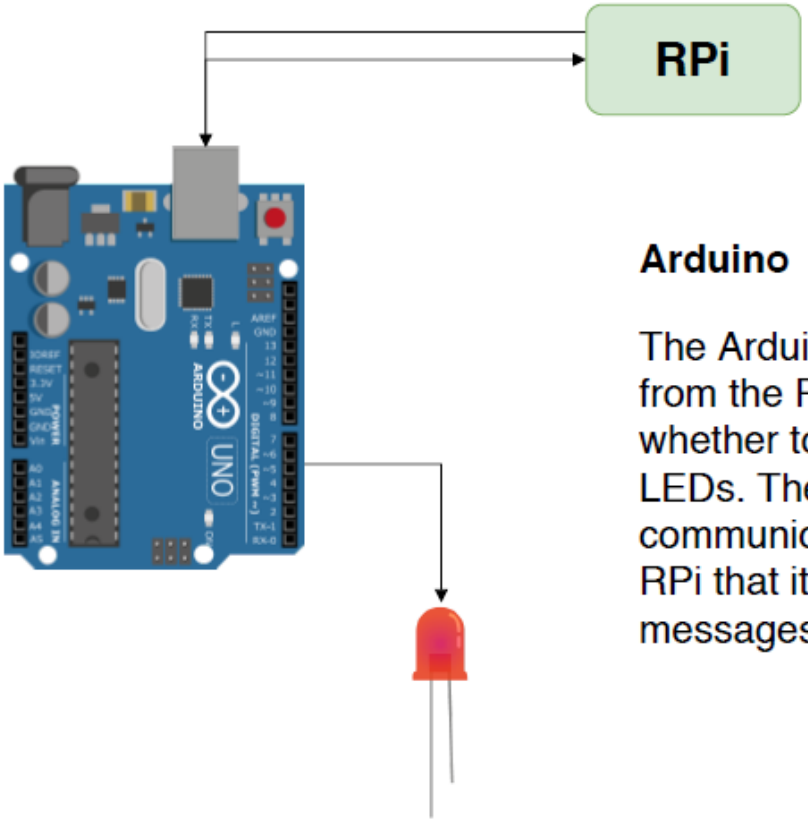
- Receive Incoming Serial data via Raspberry Pi
- Turn on/off the LED’s

Flow

The Arduino will be waiting for incoming data from the Raspberry Pi. The Arduino will take only a few commands, and the commands will determine if the LEDs need to be on or off.

When the Arduino turns on the LEDs, they will blink rapidly in random order. When the Pi calls the Arduino to turn off the LEDs, the Arduino will turn off all the LEDs.

Flowchart (Visual)



Arduino

The Arduino takes Serial data from the RPi, and determines whether to turn on/off the LEDs. The Arduino will also communicate back to tell the RPi that it acknowledged the messages over Serial.