Oregon Institute of Technology

Automated Smart Blinds Plan

# 

# 

*Authored by*:

Andrew Deraita

Michael Roberts

Chad Revel

# Table of Contents

[Table of Contents 1](#_Toc27044518)

[Hierarchical Design Diagram 2](#_Toc27044519)

[Diagram Explanation 2](#_Toc27044520)

[Sub-Modules 3](#_Toc27044521)

[Processor: 3](#_Toc27044522)

[Power Supply: 8](#_Toc27044523)

[LED Indicators: 10](#_Toc27044524)

[Up/Down Motor: 12](#_Toc27044525)

[Open/Close Motor: 16](#_Toc27044526)

[Piezo Speaker: 18](#_Toc27044527)

[Proximity Sensor: 20](#_Toc27044528)

[Temperature Sensor: 22](#_Toc27044529)

[Smoke/Gas Sensor: 25](#_Toc27044530)

[Camera: 27](#_Toc27044531)

[Wi-Fi Module: 30](#_Toc27044532)

[Front End User Application (Tier 1) 30](#_Toc27044533)

[Web Server (Tier 2) 32](#_Toc27044534)

[Web Services Software (Tier 2) 33](#_Toc27044535)

[Database Interface (Tier 3) 34](#_Toc27044536)

[MCU Wi-Fi Hardware (tier 4) 36](#_Toc27044537)

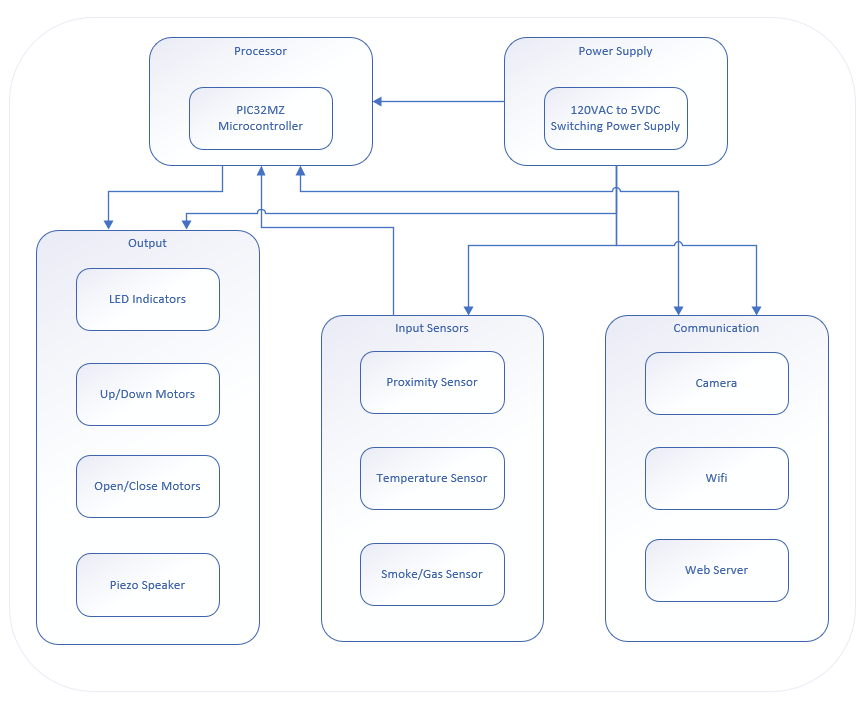
[Parts List 44](#_Toc27044538)

[Preliminary Cost 46](#_Toc27044539)

[Team Assignment 47](#_Toc27044540)

[References 48](#_Toc27044541)

## Hierarchical Design Diagram

****

### Diagram Explanation

The hierarchical diagram shows each module split apart into their own sections, which allows more of an idea on how each part is going to connect/communicate with each other. The first part of the device will be the PIC32 processor. This is the brains of the device and is responsible for communicating between each part of the entire device. The power supply will supply the necessary power to each part of the device for it to function properly. This needs to be connected to each part of the device, but some devices will require a different voltage compared to another part of the device. Example would be the LED needing 5V, compared to the Temperature Sensor which will need 3.3V.

The next part of the diagram are the output devices. These include the LED, the Stepper-motors, and the Piezo Buzzer. Each of these parts will need to receive information from the PIC32 and execute the commands from the PIC32 accordingly.

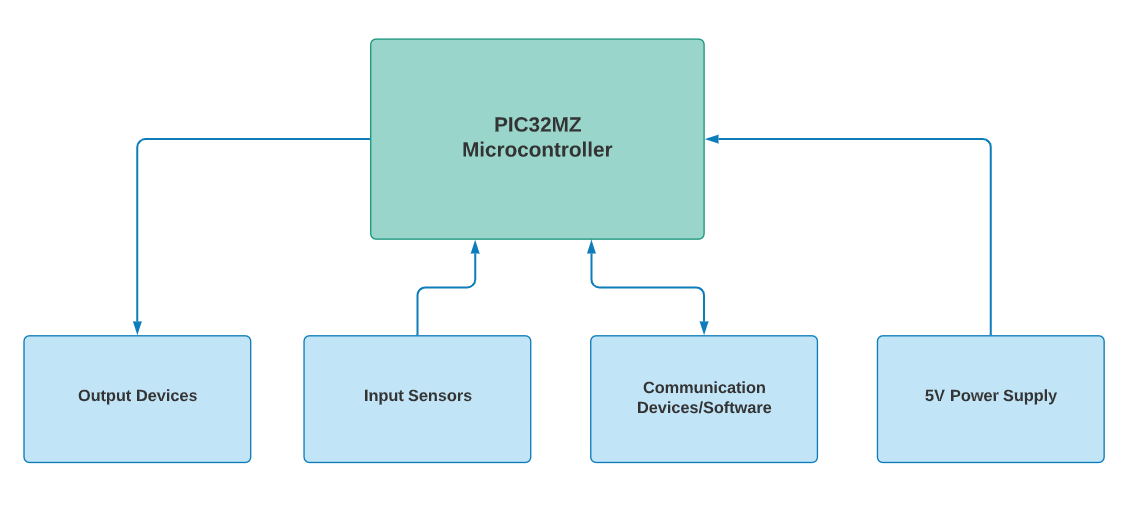
The second to last part of the diagram will be the input devices. These include the Proximity Sensor, Temperature Sensor, and Smoke/Gas Sensor. These parts will be sending a constant stream of data to the PIC32 to process.

The final part of the diagram are the Communication devices. These include the Camera, Wi-Fi, and Web Server. These devices are more unique compared to the other two parts as they need to receive and send information to the PIC32.

# Sub-Modules

### Processor:

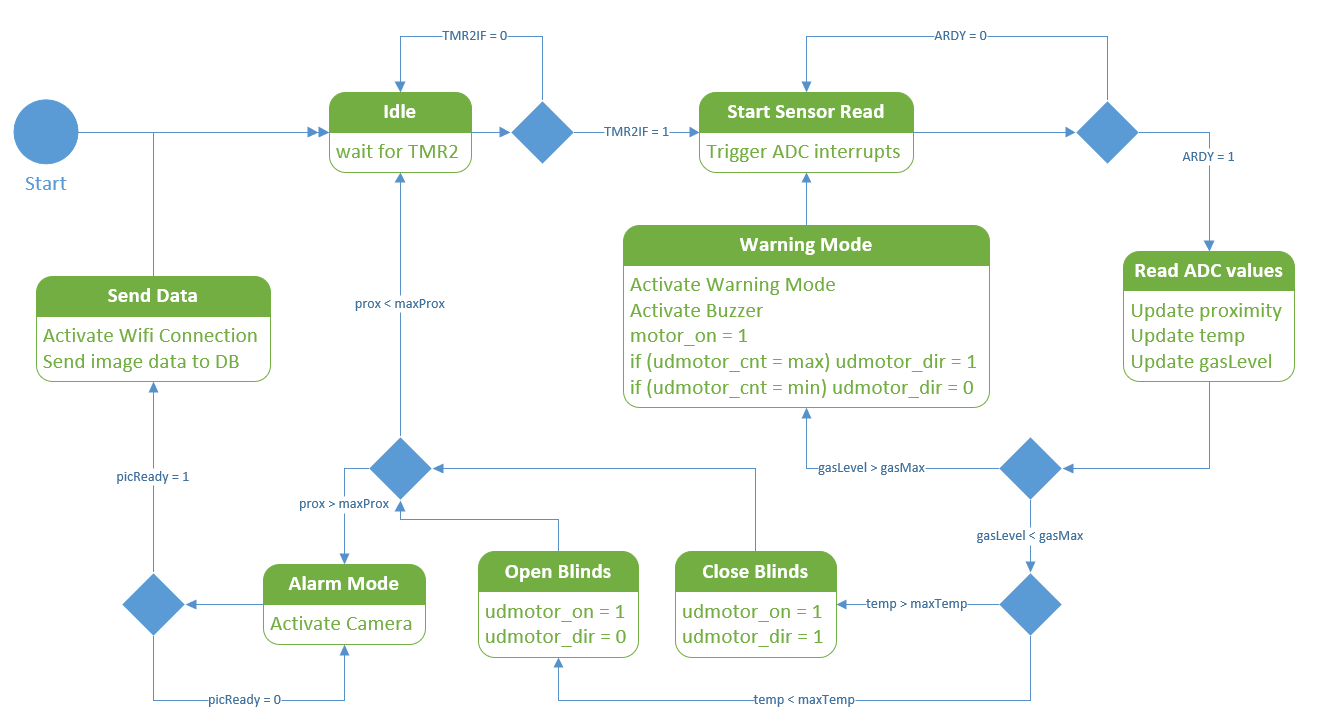
The microprocessor used to control this device will be the PIC32MZ2048EFG144 on the Olimex PIC32-HMZ144 development board. It will be configured to run at a system clock speed of 84Mhz (1 wait state). The peripheral bus clocks 2 and 3 will be utilized at 84Mhz, with further division within the required sub-module configuration.



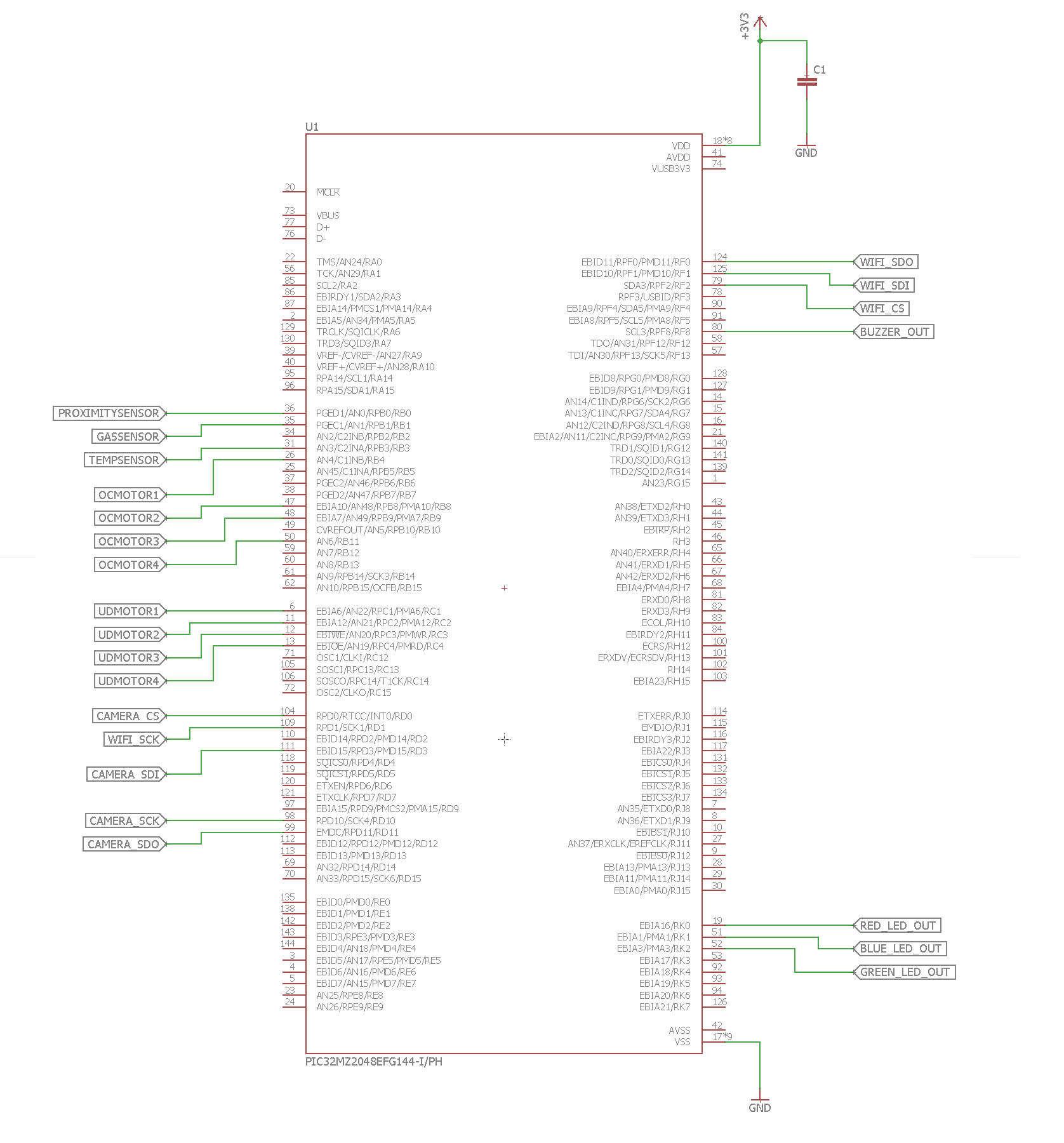
Most of the program for this device will rely on an RTS loop, coupled with interrupt driven SPI and I2C communication. To this end, the PIC32 will be set to multi-vector mode with each ISR being coded for its intended task.

The RTS loop of this program will consist of a 32-bit TMR2 interrupt that will trigger every 0.5 seconds. This ISR will trigger the ADC devices to capture the input data from the proximity, temperature, and gas/smoke sensors and update the levels accordingly.

This RTS loop will have the following state machine:



Currently, this TMR2 overflow is set to priority 1 while the camera and Wi-Fi modules will take priority at levels 4 and 5, respectively. This component will run on 3.3V and is wired up to its sub-modules according to the following schematic.



The microprocessor will have a total of 5 phases of tests that it will go through during the project’s timeline. The phases are as follows: Phase 1, Phase 2, Phase 3, Phase 4, and Final Phase.

**Phase 1:** Since the PIC32MZ comes on a development board, it has all the necessary circuitry to power on straight out of the box. Therefore, the initial test for this module will consist of applying 5V via USB cable to verify the board power on via onboard LED power indicator.

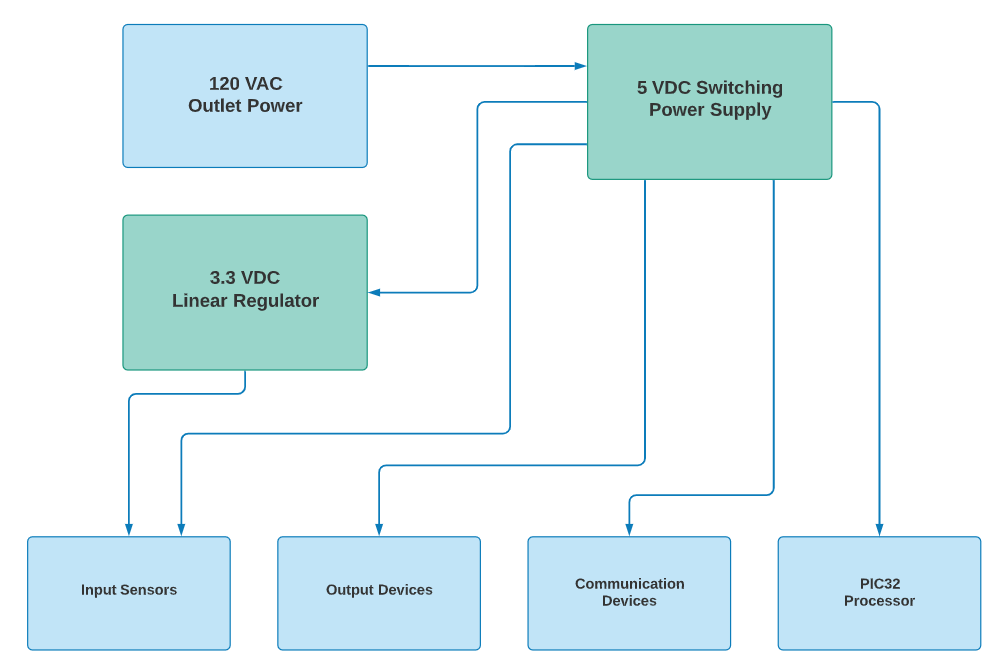
**Phase 2:** Once the board has been verified to work, a program must be written to the PIC with basic settings to verify that the processor operates as specified in the data sheet. For this test, a program will initialize the system clock and peripheral bus clock 3, as well as timer interrupts with vectors for timers 2 and 3.

**Phase 3:** Since the PIC32MZ is operating at this point, sub-modules can be implemented and added to the project. Sub-modules will be implemented in separate projects, brought through their individual testing, then integrated into the existing project. This step will be repeated for each sub-module.

**Phase 4:** Once a sub-module is ready to integrate with the project, it will be added, then each individual module will go through their final testing again, to ensure no conflicts between modules. This regression approach guarantees problems will be found and rectified as early as possible.

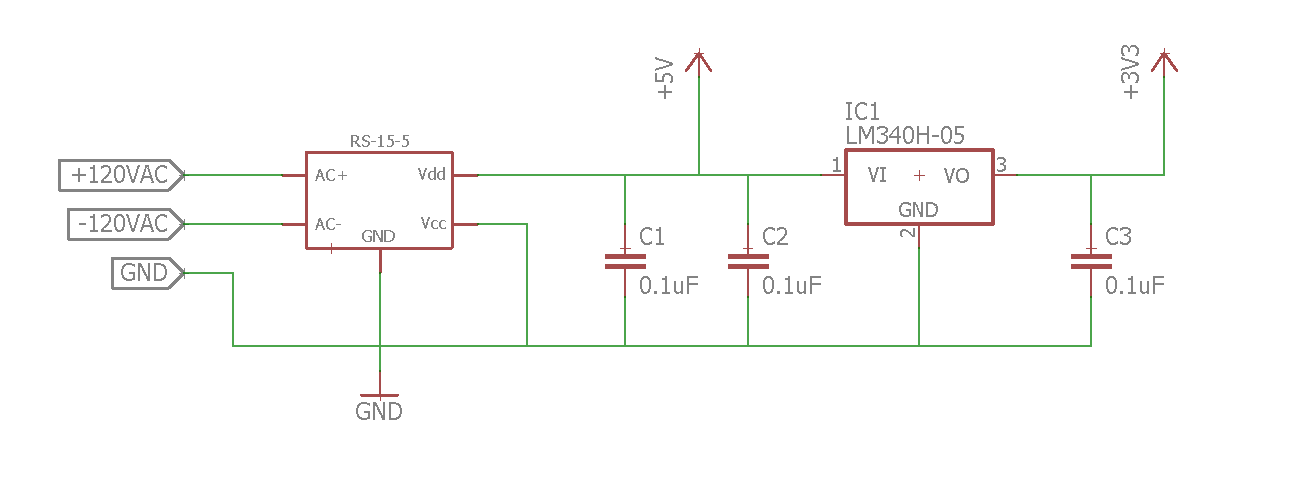
**Final Phase:** Once all modules are added, the unit will be installed on a test blind and setup to run through a final test to verify each module, and to verify consistency in operation.

### Power Supply:



The power for this device will be supplied initially with a MeanWell RS-15-5 switching power supply. This will convert the 120VAC from a typical home outlet to 5V, with a maximum current of 3A.

The LM340 linear voltage regulator will be used to further reduce the voltage to accommodate certain modules, such as the temperature sensor, Wi-Fi module, and PIC32MZ to their voltage requirements. The wiring diagram below will show on how the power supply will be hooked up to the rest of the system.



The power supply will have a total of 5 phases of tests that it will go through during the project’s timeline. The phases are as follows: Phase 1, Phase 2, Phase 3, Phase 4, and Final Phase.

**Phase 1:** The unit will be installed with a 120VAC capable power cord with an appropriate US 3-prong power plug. Once plugged in, a voltmeter is used to verify the unit is supplying the specified 5V.

**Phase 2:** Once the unit is confirmed to the specified output voltage, a dummy load consisting of a high-power resistor will be applied. The power across this resistor can be measured, thus confirming the supply module can output the correct current at its specified voltage.

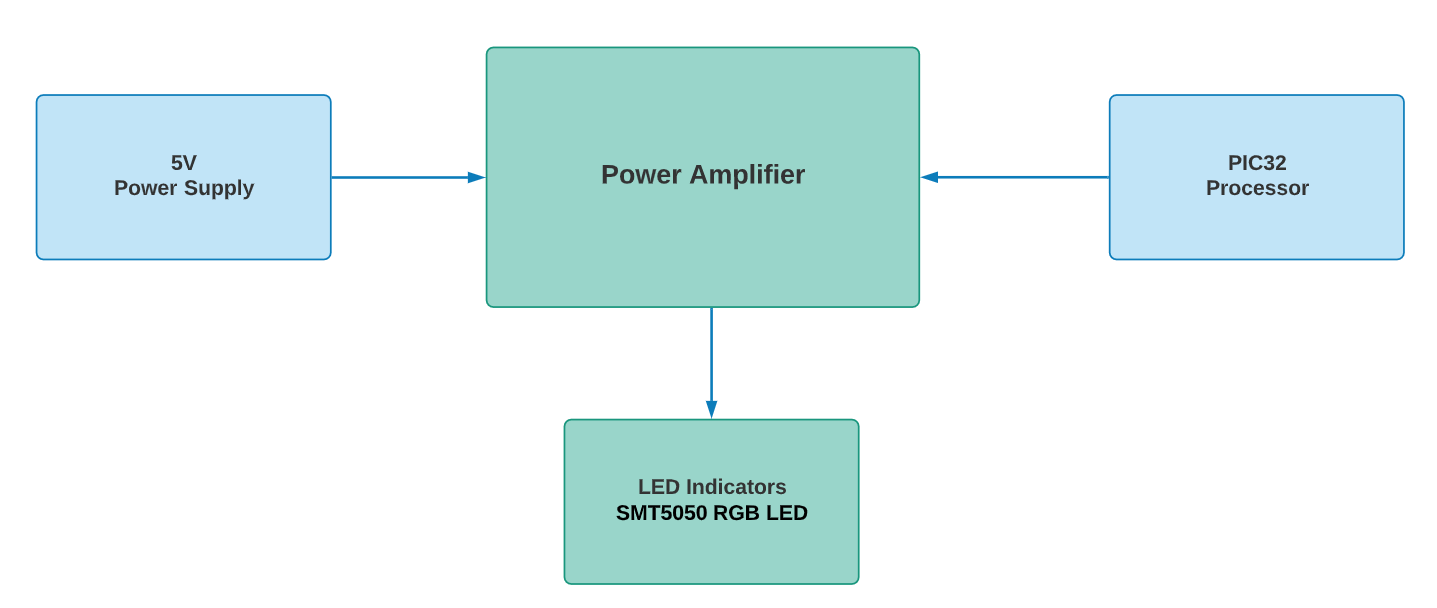
**Phase 3:** This phase consists of building the remaining power circuitry and testing it. The soothing capacitors and 3.3V regulator network will be implemented and verified via voltmeter, and further dummy load testing on the 3.3V rail.

**Phase 4:** Each sub-module will require power as it is added to the project. As these sub-modules are added, the load of the power supply will be measured under the most demanding circumstances to verify that the total current output of the 5V supply does not exceed its maximum current of 3A.

**Final Phase:**  Once all sub-modules are installed and deemed operational, a situation will be constructed that will draw the most power possible. The system will be held in this state to simulate a possible error scenario. Power draw from the 5V supply will be measured to guarantee no critical power failures.

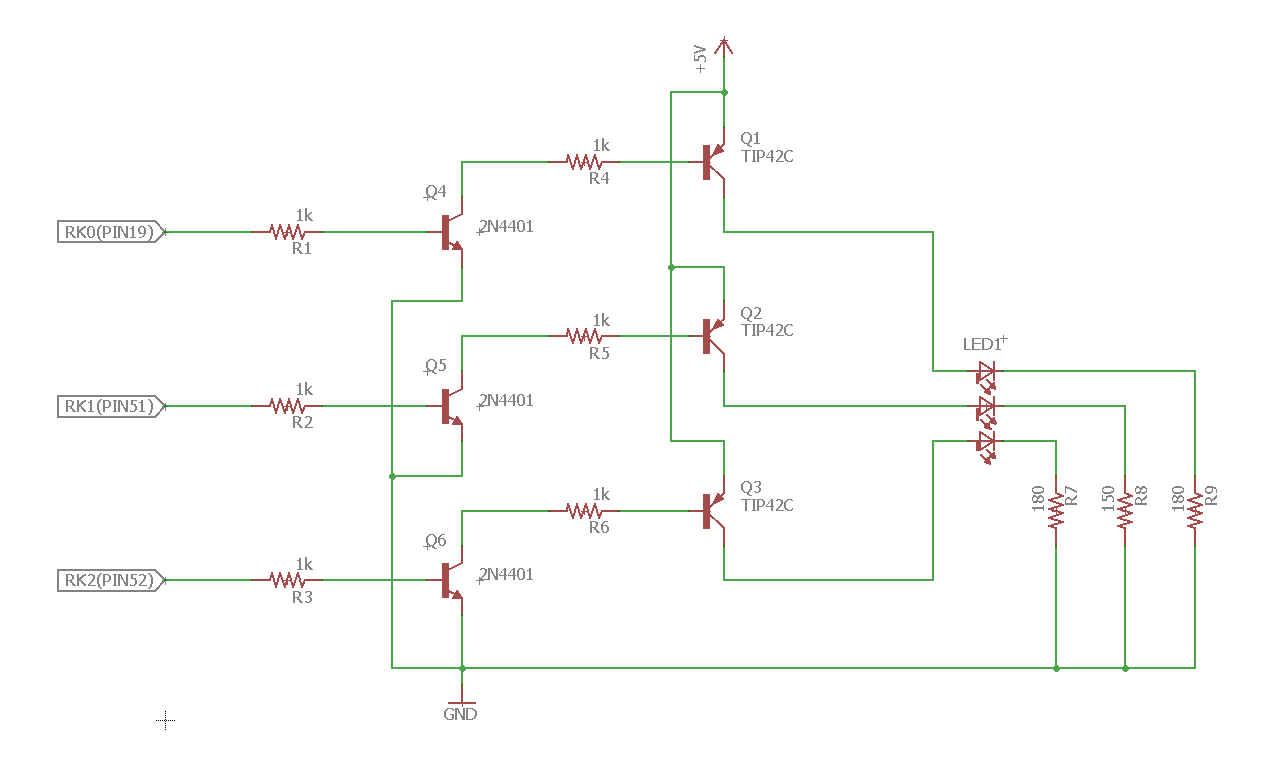
### LED Indicators:

The LED is a single SMT5050 RGB LED and will be a necessary part of the proximity sensor sub-module. It will be used in tangent with the “alert mode” to notify nearby individuals of operating conditions. This LED contains a red, green, and blue dye, and can combine these dyes to produce any color combination required.



Each dye of the LED will be wired to an output of the PORTK register and controlled with software. RK0 will control the red dye, RK1 will control the blue, and RK2 will control the green dye. The bits must be set as digital outputs in the PIC32MZ configuration code. The green color indicates normal operation and will power only the green dye. Similarly, the red warning indicator will use just the red dye. However, when the device requires an orange light, both the red and green dyes will be activated. Since these dyes are close enough together, the light will combine and appear orange.

These output pins will be controlled by the TMR2 ISR when updated values from the proximity sensor are read. Cutoff levels determined by the proximity sensor reactivity chart will be used to determine which LED’s to activate. Since these LED’s pull more current than the PIC32MZ processor output pins can supply, a power amplification circuit will be used to power the LED’s directly from the 5V line.



The LED will have a total of 4 phases of tests that it will go through during the project’s timeline. The phases are as follows: Phase 1, Phase 2, Phase 3 and Final Phase.

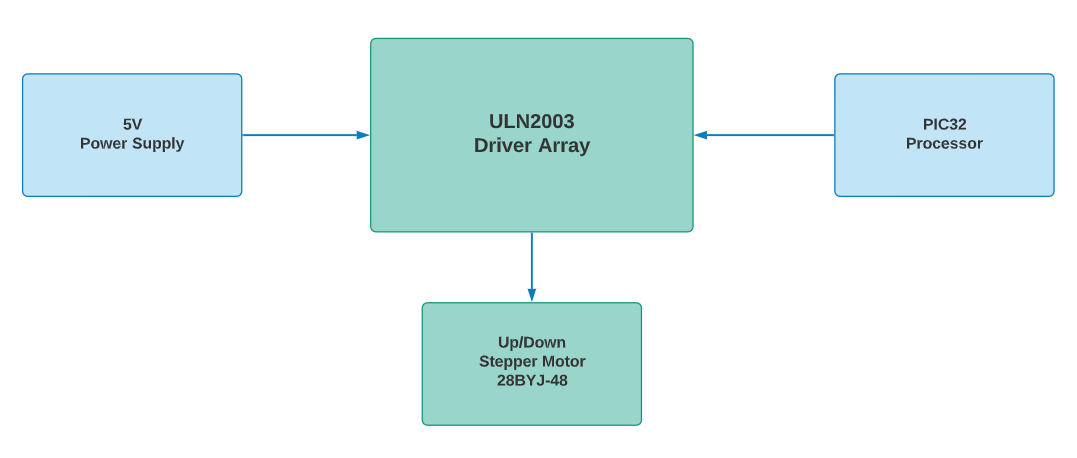
**Phase 1:** To verify the LED’s illuminate as specified, a dummy 5V circuit will be created for each dye, with individual current limiting resistors.

**Phase 2:** The power amplification circuit will be built and tested. LED’s should illuminate with full power and intensity with 3.3V applied to their outputs. A multimeter will be used to verify the LED’s load is coming directly from the power supply, rather than the PIC32 output pins.

**Phase 3:** All LED dyes will be illuminated at full power and ran for 24 hours. This will verify that they are not drawing too much current or have internal failures not identified in the initial testing phase.

**Final Phase:** The final phase will be to integrate the LED into the rest of the device. Once this is done, then it will go through regression tests with the rest of the sub-modules to eliminate any final issues that occur.

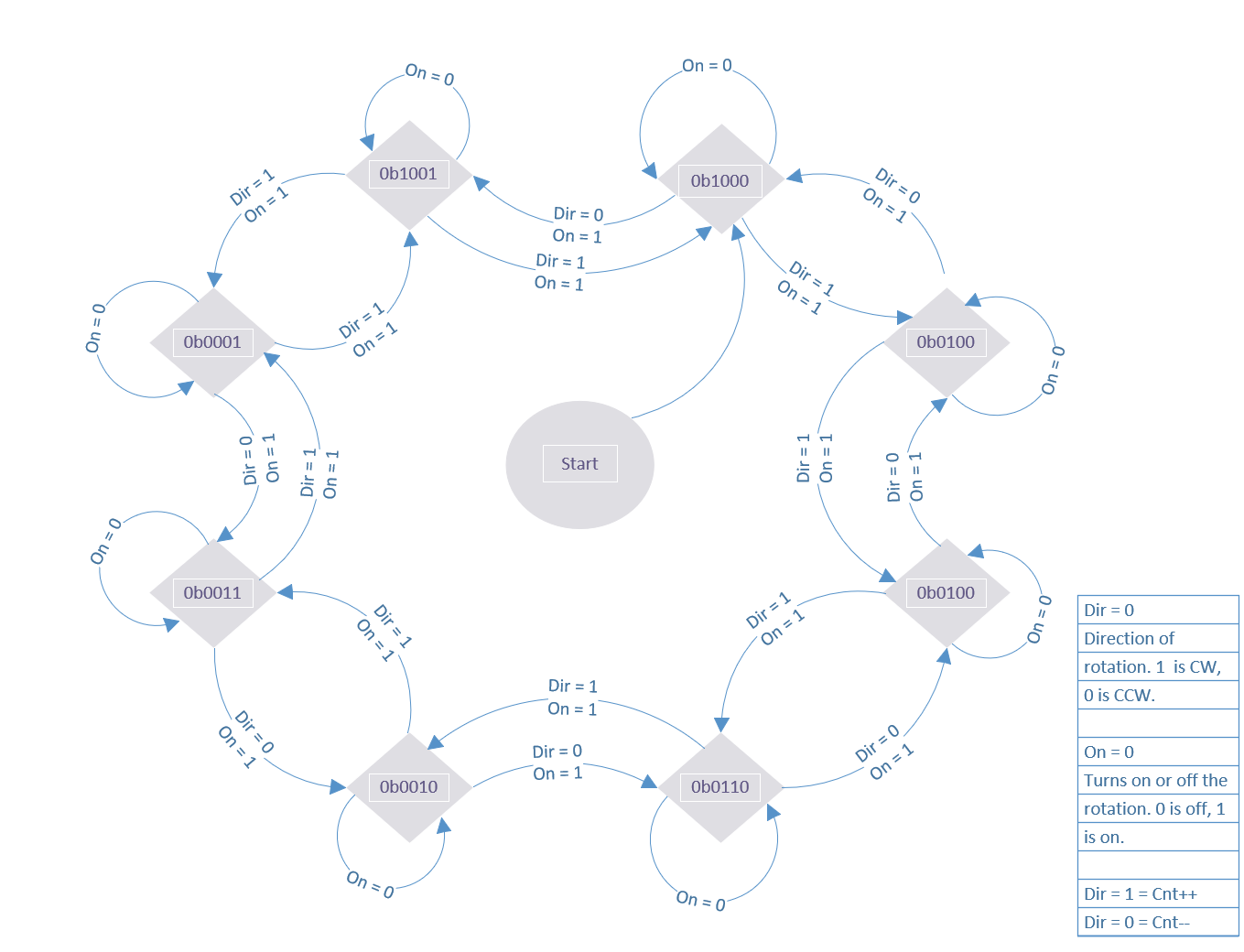
### Up/Down Motor:



The motor(s) that will be used for the device is the Elegoo 28BYJ-48 ULN2003 5V Stepper Motor. They also come with the ULN2003 Driver board. This specific motor will be used for raising and lowering the blinds by a string attached to a spool. The motor will require 5V to operate at optimal capacity. The power will be going through the driver array, which will be the hub for both the power and the PIC32 that will operate the motor. The Driver Array will be the primary source for power from the power supply and signal processing from the PIC32MZ.

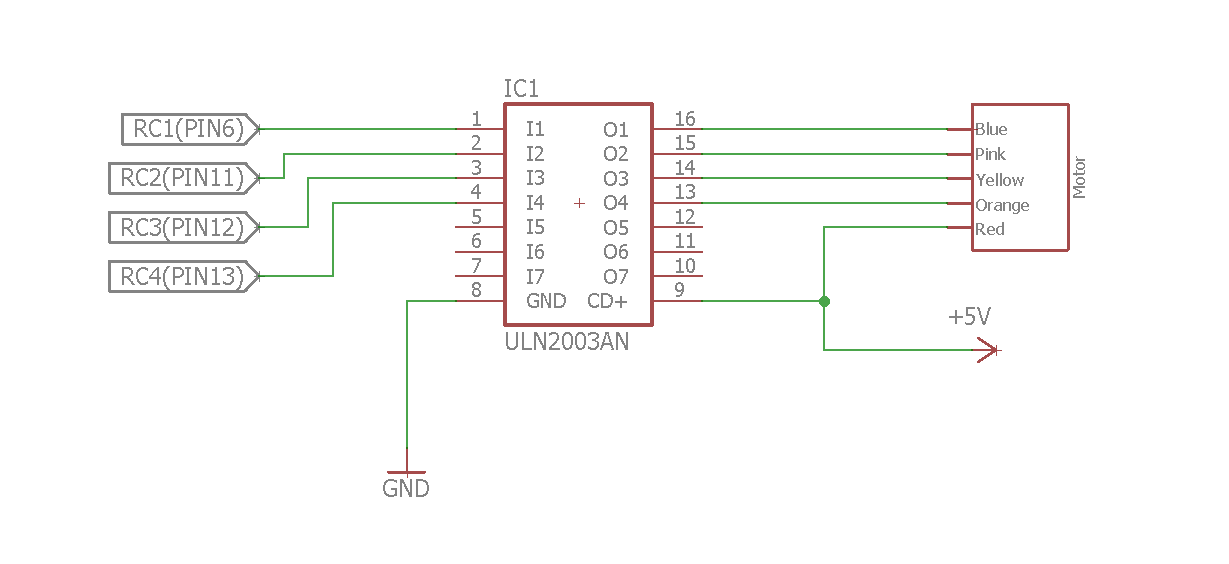
Since these motors are 513 steps per revolution (0.7 degrees per step), the motor can be operated in full step mode without any further accuracy needed. The motor inputs will be controlled via encoding in the PIC32MZ. This will be done by sending signals in the following pattern to the Driver Array: 0b1000 → 0b1100 → 0b0100 → 0b0110 → 0b0010 → 0b0011 → 0b0001 → 0b1001.

This pattern, when sequenced 64 times will cause the motor to turn one full revolution. Once this pattern of signals has completed one cycle, a counter variable will either increment up or down by one. Allowing the system to keep track of the exact position the blinds are at. This can be seen in the signal diagram below.



The wiring diagram shown below explains which of the pins will be connected the Driver Array, and what pins the Array will be connected to the motor. One of the differences between the Up/Down and the Open/Close motors is that they’ll have their own set of 4 registers that are solely responsible for each motor. In this case, we’ll be using RC1, 2, 3, and 4 via Pins 6, 11, 12, and 13 to connect to the Control Array. Furthermore, TMR5 will be used to control the speed at which the motor will rotate. It will be set to a 16-bit configuration with a 1:1 pre-scalar.

The Control Array and the Motor will need to be hooked up as seen in the wiring diagram below. Example would be of RC1 is connected to the first port of the Control Array of IC1. Which will then have O1 of the Array connect to the blue wire of the motor.



The Up/Down Motor will have a total of 4 phases of tests that it will go through during the project’s timeline. The phases are as follows: Phase 1, Phase 2, Phase 3, and Final Phase.

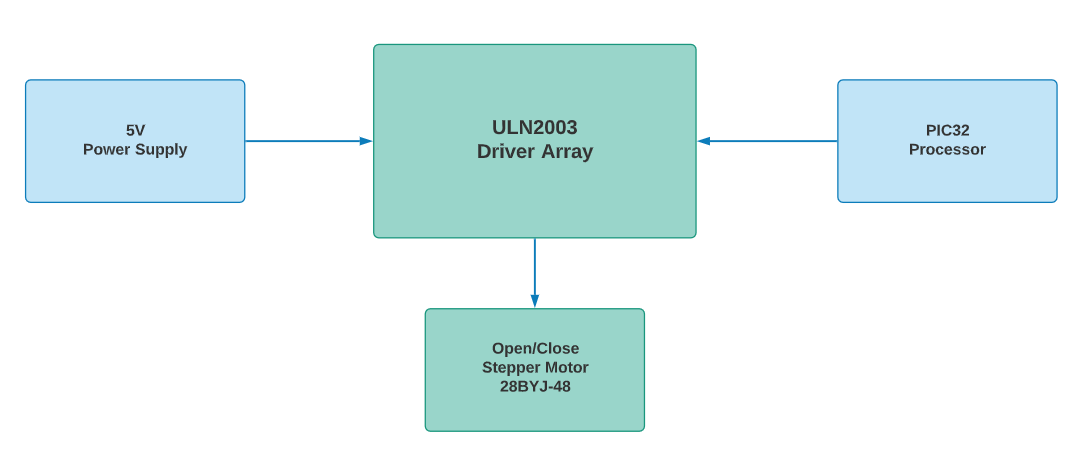
**Phase 1:** For the initial test, the stepper motors will be attached to their driver array and to a 5V power supply. It will go through a basic rotation function to make sure that the motors will properly rotate as intended.

**Phase 2:** The next phase will be to implement functions that will save settings for which position that will be used. Examples of these settings are full up, full down, ¼ up, ⅓ up, ½ up, ⅔ up, and ¾ up. Also, might have to add extra variable names for the various positions the motors will remember in the settings. Examples being ¼ up is also ¾ down.

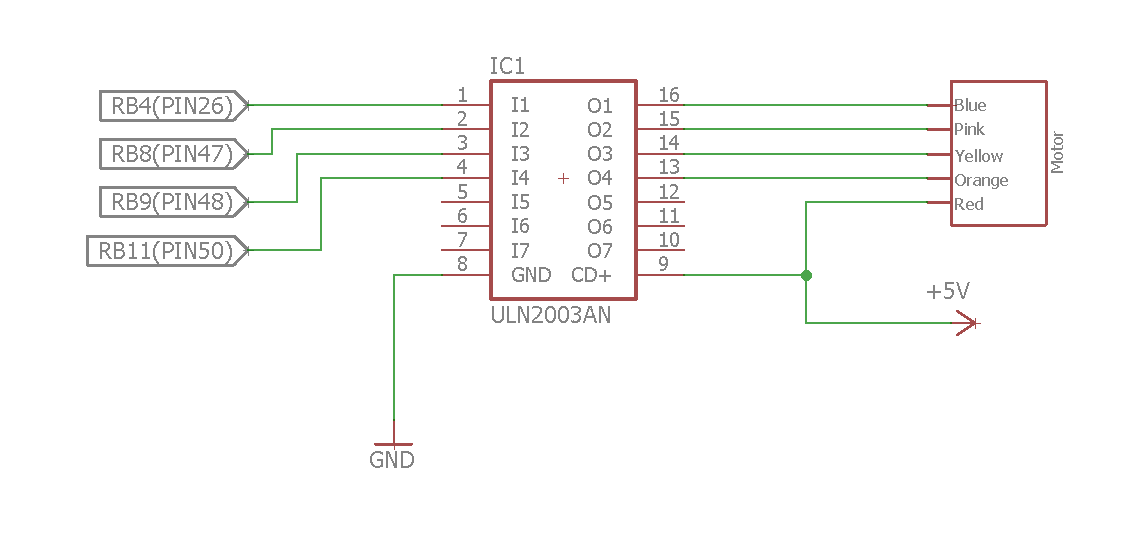
**Phase 3:** This step will be to incorporate at least two separate parts together that will communicate with one another and act according to the settings set within the program. An example of this test would be if the temperature sensor detects a certain temperature, causing the blinds to close or open.

**Final Phase:** The final testing will be when this system is fully integrated into the main device along with the rest of the devices. It will go through each part that requires the blinds to either be up or down. It will follow the same regression testing that the rest of the parts will go through during this phase to eliminate and further problems.

### Open/Close Motor:



The Open/Close motor will be set up mostly the same as the Up/Down motor since they are the same Elegoo 28BYJ being used, along with the same Driver Board. This motor will also require 5V in order to operate and will need to have the same sequence of signals to loop through for the motor to full rotation. TMR5 will be used again to control the speed of the motor. It will be set to a 16-bit configuration with a 1:1 pre-scalar.

Like the previous motor, the Open/Close Motor will have its own 4 unique registers assigned to it. The 4 registers we’ll be using are RB4, 8, 9, and 11 via Pins 26, 47, 48, and 50. The registers will need to be set up in the exact order as shown in the wiring diagram below. The registers will be set up through a function that will make the motor rotate by sending the signals in the proper order. This will be done by sending signals in the following pattern: 0b1000 → 0b1100 → 0b0100 → 0b0110 → 0b0010 → 0b0011 → 0b0001 → 0b1001.

The Open/Close Motor will have a total of 4 phases of tests that it will go through during the project’s timeline. The phases are as follows: Phase 1, Phase 2, Phase 3, and Final Phase.

**Phase 1:** For the initial test, the stepper motors will be attached to their driver array and to a 5V power supply. It will go through a basic rotation function to make sure that the motors will properly rotate as intended.

**Phase 2:** The next phase will be to implement functions that will save settings for which position that will be used. Examples of these settings are full open, full close, ¼ open, ⅓ open, ½ open, ⅔ open, and ¾ open.

**Phase 3:** This step will be to incorporate at least two separate parts together that will communicate with one another and act according to the settings set within the program. Example is for the blinds to fully open when the Gas/Smoke Sensor detects smoke.

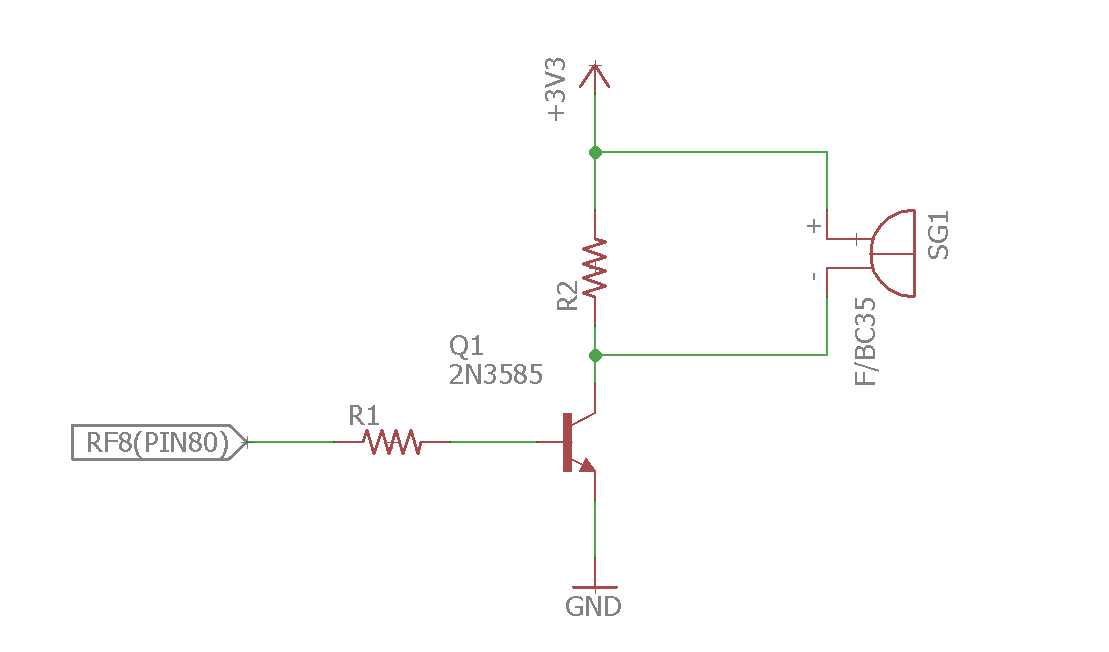
**Final Phase:** The final testing will be when this system is fully integrated into the main device along with the rest of the devices. It will go through each part that requires the blinds to either be opened or closed. It will follow the same regression testing that the rest of the parts will go through during this phase to help eliminate any unforeseen issues.

### Piezo Speaker:



The speaker used for the device will be the Piezo Buzzer PS1240. The Piezo Buzzer will have the loudest tones when used around 4KHz which is around 70dB. The Buzzer will require a 3.3V power supply in order to function properly.

The buzzer will be hooked up to the register RF8 through PIN80 on the microprocessor. Additionally, in order to produce the proper dB of sound, we’ll need to have TMR4 set to 1/4KHz for the correct timing. The wiring diagram below will show on how to set up the buzzer to the board and the processor.



There Buzzer will have a total of 4 phases of tests that it will go through during the project’s timeline. The phases are as follows: Phase 1, Phase 2, Phase 3, and Final Phase.

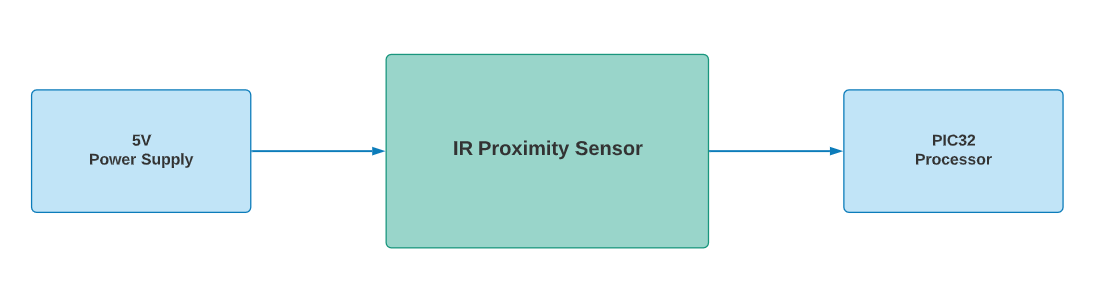
**Phase 1:** The Piezo speaker will be one of the few pieces that will require the 3.3V compared to the 5V in order to function properly. This will be tested with a multimeter to make sure that the speaker has connectivity. After the initial test is complete, the buzzer will be hooked up to a board and be tested that it will generate sound.

**Phase 2:** For the next phase, it will be wired to the PIC32 and be tested to have the proper dB by using TMR4 being set to 1/4KHz.

**Phase 3:** The next step will be to test if it works in conjunction with another part of the project. An example would be to use the gas sensor alongside the buzzer. If the gas sensor detects smoke then the buzzer will activate, and when the sensor no longer detects smoke, then the buzzer will go silent.

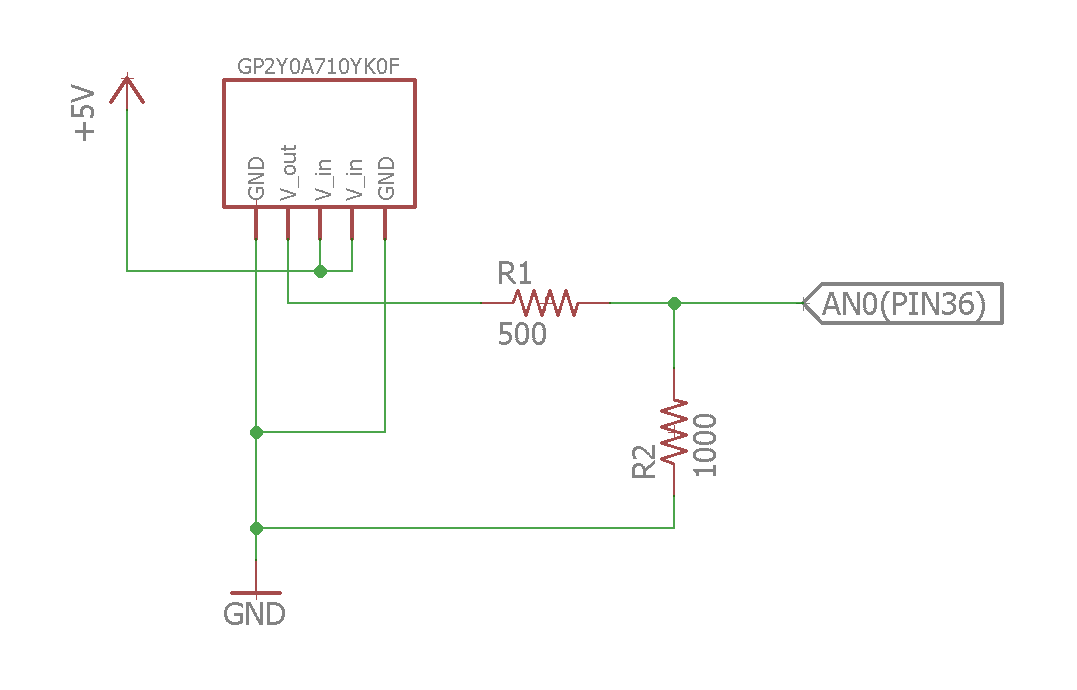
**Final Phase:** For the final testing, the speaker will be implemented with the remaining pieces of the device and go through various tests to make sure that each piece is able to function properly with one another. An example of this will be to use the proximity sensor along with the LEDs, speaker, and the blinds if there is a moving object within a certain range of the sensor. This sub-module will be going through the same regression testing that the rest of the system will be going through.

### Proximity Sensor:



The Sharp GP2Y0A710YK0F Long Range IR Distance Sensor will be used for the security part of the project and will need 5V to operate. This unit outputs an analog voltage, relative to the distance the motion was detected. Therefore, the ADC0 unit is needed on the PIC32MZ. This unit will divide off PBCLK2 to have a 21Mhz sample rate, as specified in the ADC specifications to ensure enough time has elapsed for the internal RC pump to charge. This module will connect directly to the PIC32 via PIN36 (AN0) and be triggered by the RTS loop to read motion distance.

A distance of 300cm will serve as the cutoff point for acceptable motion, and the processor will tell the LED to turn yellow. If the target of the motion gets within 225cm, this will indicate to the processor that the target is too close. It will then activate the camera and shutter the blinds. These levels of motion detection will be set initially in software, but the user will be able to change them through web interface commands. The wiring diagram below shows how the sub-module should be wired up to the system.



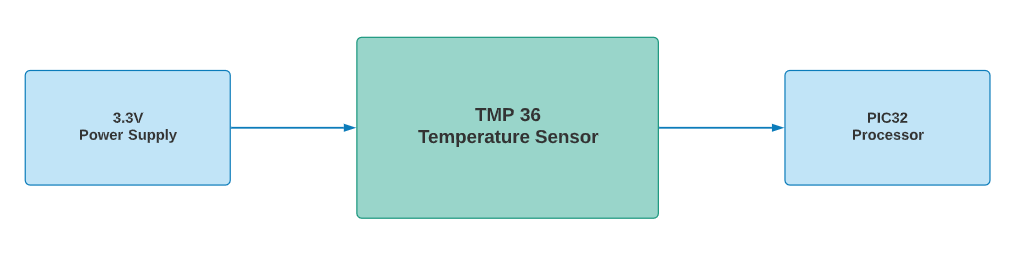
The proximity sensor will have a total of 3 phases of tests that it will go through during the project’s timeline. The phases are as follows: Phase 1, Phase 2, and Final Phase.

**Phase 1:** This unit will need to be powered directly from the 5V supply line. Once installed, a multimeter will be used to probe the analog output to verify the reading changes based on movement in front of the lens.

**Phase 2:** In this phase the proximity sensor will be attached to the AN0 analog pin of the PIC32 and software will be written to interpret the ADC signal into distances according to the reaction charts in the sensor datasheet.

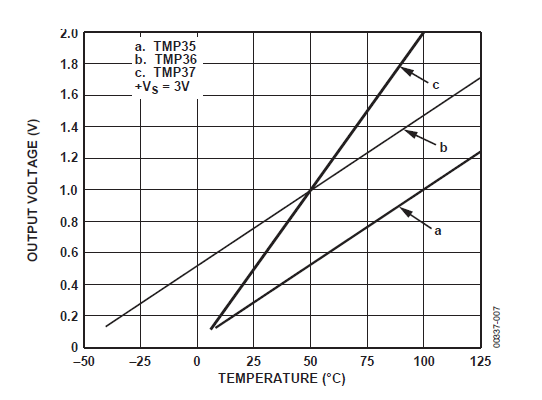
**Final Phase:** This unit will then be integrated into the main project and set to trigger the camera, LED’s and motors according to objects detected at appropriate distances. As this unit is integral to the operation of several units, successful regression testing will be required for the previously mentioned sub-modules to be at least in their first phase of testing to verify functionality.

### Temperature Sensor:

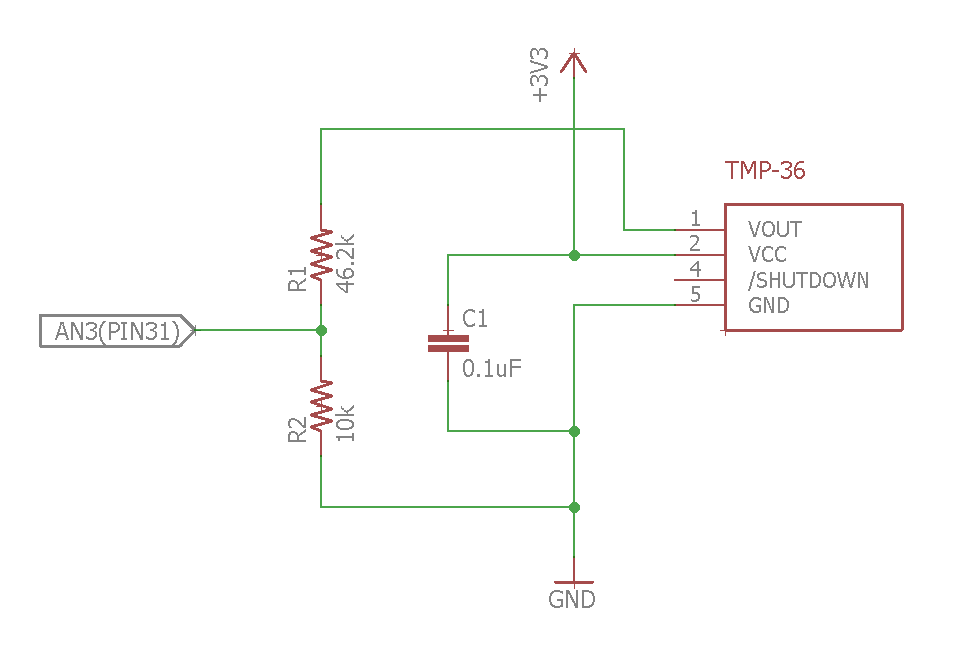


The temperature sensor used for this device will be the TMP36 Analog Temperature Sensor. The TMP36 device is intended for a single supply operation from 2.7V to 5.5V, with a supply current below 50 μA. Due to PIC32MZ only being a 3.3V chip, this device will be powered at the same level to avoid input attenuation issues.

This device will be supplying an analog voltage to the ADC3 unit inside the PIC32MZ via PIN31 (AN3) and will be measured during the RTS loop described previously. The ADC unit will capture the analog voltage of the output pin with 12-bit resolution, allowing the microprocessor to calculate the temperature according to line “b” on the following chart.



An additional voltage divider circuit is required to convert the Celsius temperatures from this chart into Fahrenheit, with a 58-degree offset. Therefore, the above numbers will be converted to their Fahrenheit equivalent and adjusted in software. The wiring diagram below will show how the sensor will be set up, and how we’ll achieve the conversion for Celsius and Fahrenheit.



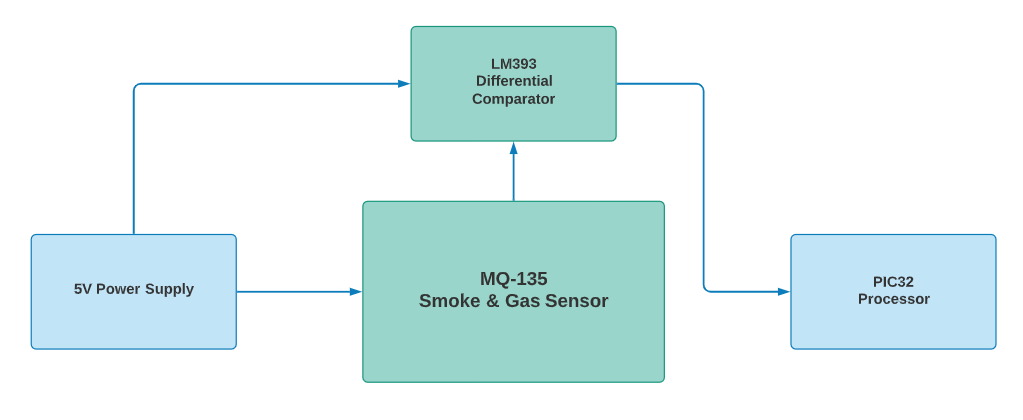
The temperature sensor will have a total of 3 phases of tests that it will go through during the project’s timeline. The phases are as follows: Phase 1, Phase 2, and Final Phase.

**Phase 1:** The temperature sensor will be drawing power from the 5V power supply and sending information to the PIC32 about the surrounding ambient temperature. For the initial test, it will be checked with a multimeter to make sure the device is not faulty.

**Phase 2:** For the next test, the temperature sensor and the stepper motor will be wired up to the PIC32 and ran through a basic program that will trigger the motor to rotate once the temperature reaches a set degree. The motor will then stop rotating once the temperature drops below the set degree.

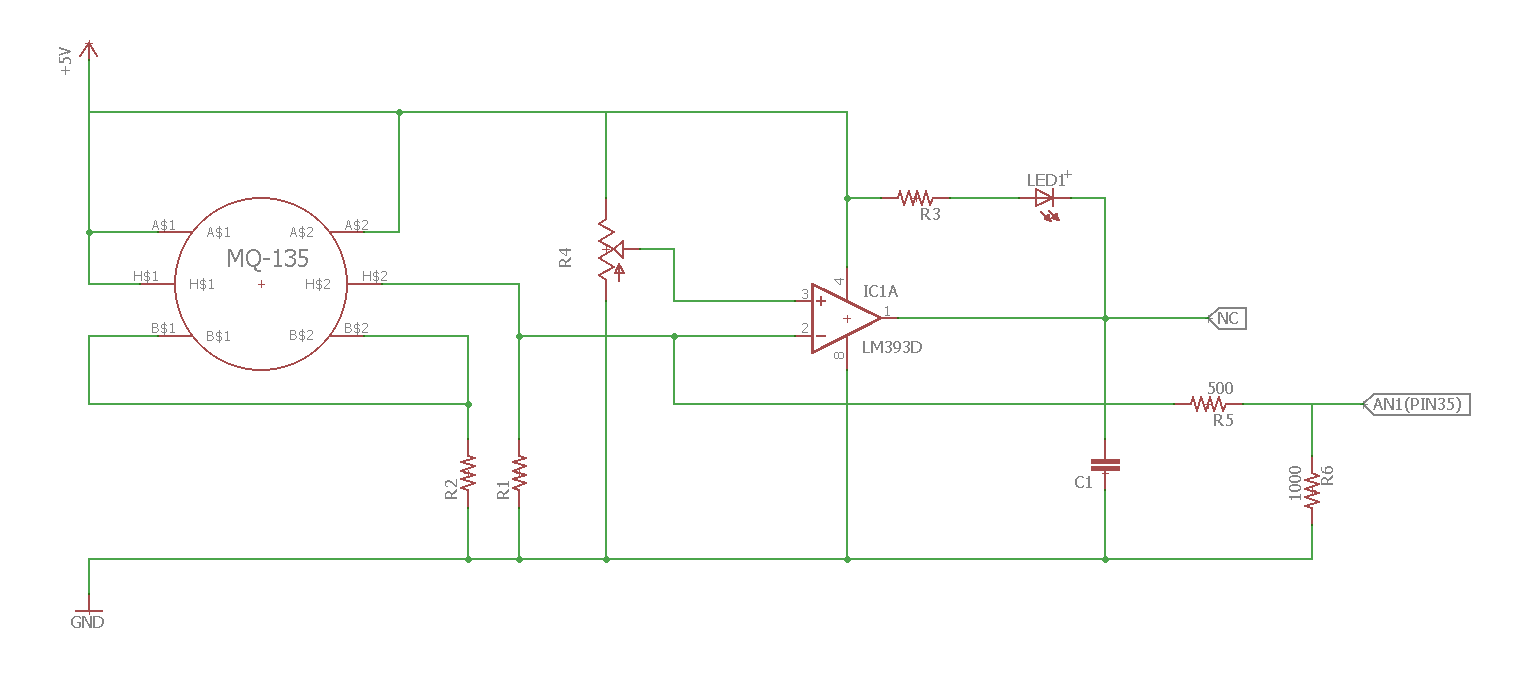
**Final Phase:** Once Phase 2 is completed, the temperature sensor will be incorporated with the rest of the device. Once integrated to the rest of the system, regression tests of the entire system will commence to eliminate any final unforeseen issues.

### Smoke/Gas Sensor:



This sub-module will use the MQ-135 Air Quality Sensor Hazardous Gas Detection Sensor. This device has a small form factor and can detect dangerous levels of NH3, NOx, alcohol, Benzene, smoke, CO2 and others forms of gas. When contaminated gas is present in the environment, the conductivity of the sensor increases as the concentration of contaminated gas in the air increases. This is fed into a LM393 comparator that outputs a variable voltage, depending on the concentration of detected gasses.

The setup for the gas sensor will have 2 parts: the sensor itself, and the LM393 Differential Comparator. While the sensor itself will send info to the comparator, the comparator will then send the info to the PIC32 processor. The gas sensor will be using 5V for power and will be hooked up to the ADC1 module via the PIN35 (AN1).



The Smoke/Gas Sensor will have a total of 4 phases of tests that it will go through during the project’s timeline. The phases are as follows: Phase 1, Phase 2, Phase 3, and Final Phase.

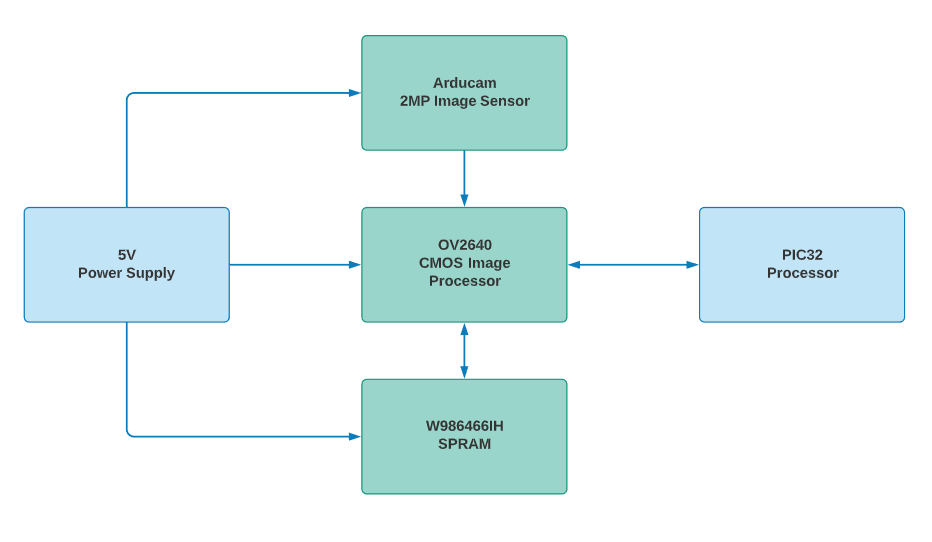
**Phase 1:** This unit is powered directly with 5V, so it can be tested using the power supply sub-module. A multimeter will be used to probe the analog voltage out to measure if the value changes with the presence of smoke.

**Phase 2:** Once the unit is verified to power on and provide output readings, it will be attached to the AN1 pin of the PIC32 ADC unit to read its values. Measurements will be taken with the PIC under simulated smoke settings to verify the readings follow the expected reactivity chart of the data sheet.

**Phase 3:** The next step will be to have the gas sensor hooked up alongside with the buzzer to make sure it works together with the buzzer. So, if the gas sensor detects gas, the buzzer will activate and once the sensor no longer detects any gas then the buzzer will turn off.

**Final Phase:** Error levels of smoke will be applied to the sensor to verify the sub-module correctly triggers warning mode under expected circumstances. Once this is done, it will be incorporated with the rest of the device and go through regression tests with all sub-modules to eliminate any final issues.

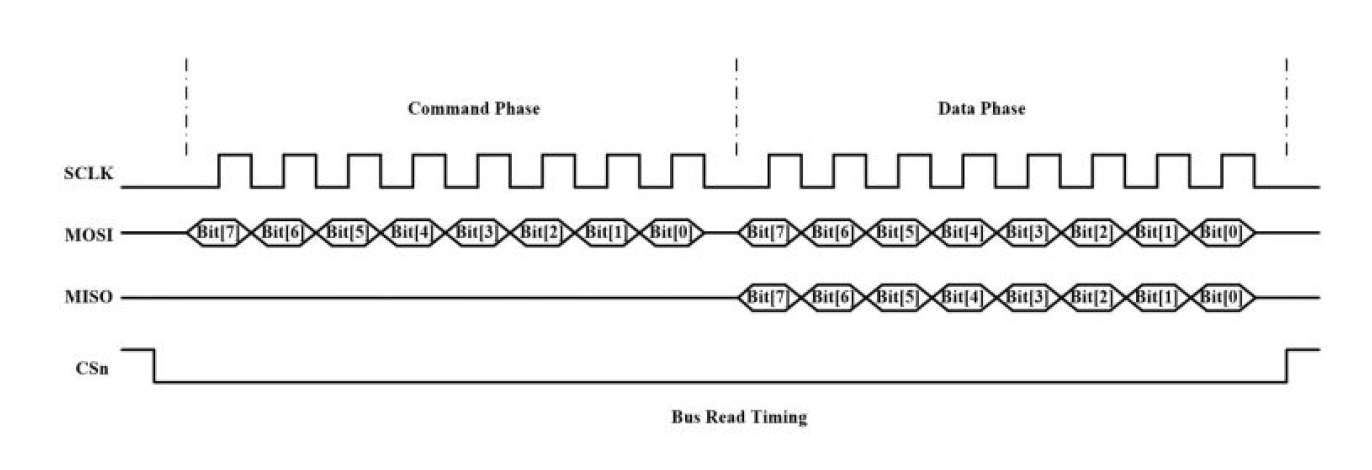
### Camera:



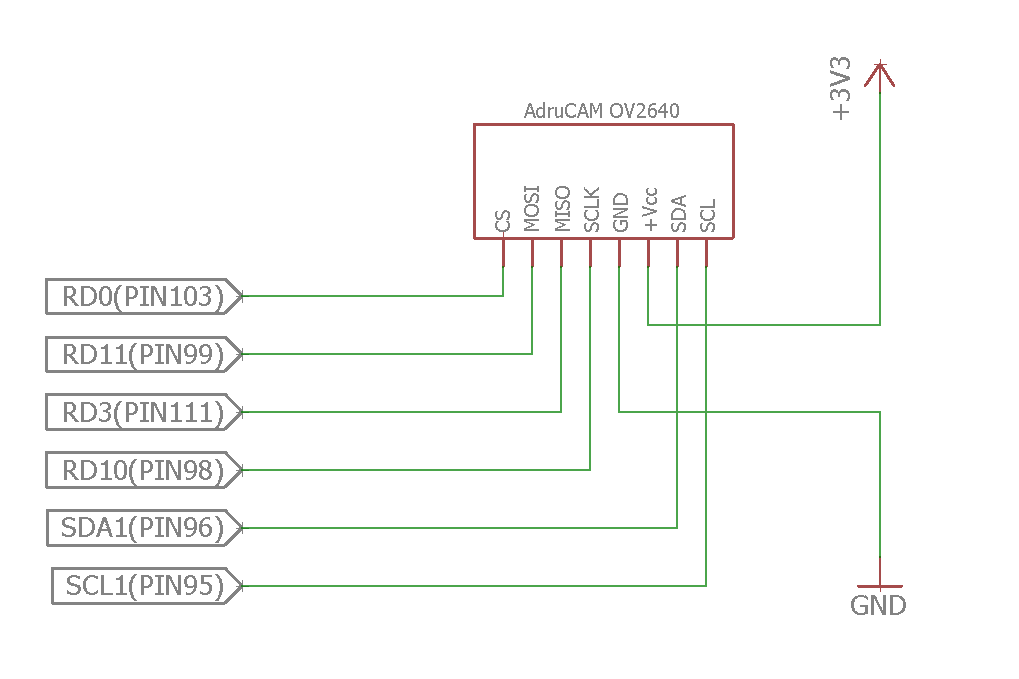
The camera sub-module consists of the Arducam Mini Module Camera Shield with OV2640 2 Megapixels Lens. This 2MP camera supports image resolutions in UXGA mode up to 1600 x 1200 pixels, which is more than enough for image recognition. To control this device, the PIC32MZ will communicate directly to the OV2640 via an SPI slave interface to transmit camera commands and data.

The SPI interface requires a clock no faster than 8Mhz, with a minimum data setup time of 100ns and minimum hold time is 0ns. Since the SCK is going to operate at 7.63Mhz (by setting SPI4BRG to 10), we know the setup time will be Tsck/2 - 15 = 131ns - 15 = 116ns which meets the timing requirement of 100ns for the OV2640. Similarly, the hold time will be Tsck/2 + 15 = 131 + 15 = 145ns which also meets the requirements.

Configuring the OV2640 will require setting the number of frames to capture, image mode, and enabling burst FIFO reading, which is done via SPI writes to the control registers found in the OV2640 datasheet. This data will be sent to the camera according to the following timing specification:



This protocol will allow jpeg image data to be sent back to the microcontroller, which will then send the data to the web-based application software via Wi-Fi. Allowing the owner to see the images captured. This device will have multiple parts to it, and each one will need to have a 5V power supply. To start, the 2MP sensor will send its info to the CMOS image processor, and the SRAM will also send its info to the CMOS. Once the CMOS has the information from its embedded chips, it will then send that info to the PIC32. Since these chips are already integrated with the camera PCB, the only consideration will be communicating via SPI to transmit data.



The camera will have a total of 4 phases of tests that it will go through during the project’s timeline. The phases are as follows: Phase 1, Phase 2, Phase 3, and Final Phase.

**Phase 1:** This module does not have any power indicators on it, however it does have an Arduino library associated with it. To initially test this module, it will be installed on an Arduino, along with the library software, and sample code will verify the sub-module can produce images.

**Phase 2:** Now that the module is confirmed to operate as specified by its data sheet, it will be wired up to the PIC32’s SPI4 module. Software will then be implemented to send commands to the camera. Reading the status register on the chip will confirm that the sub-module is compatible and can successfully communicated with the system.

**Phase 3:** Once communication with the camera module is working, ISR routines will be written to active the camera, instruct it to take pictures, and send those pictures back to the processor. Each command will be tested to work individually before the module is considered operational.

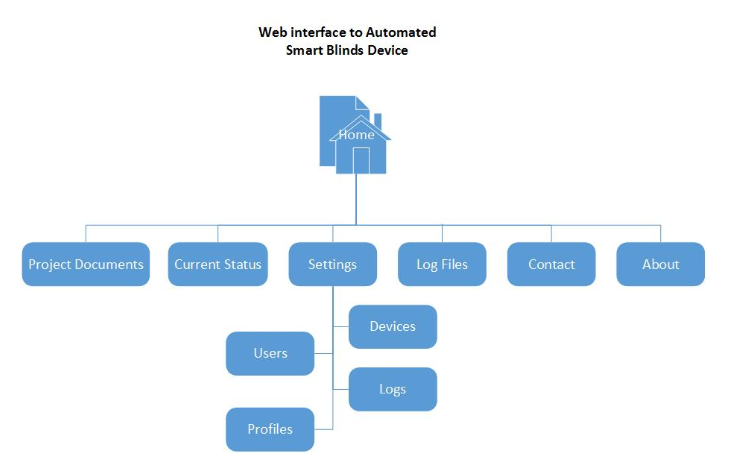
**Final Phase:** Full verification of this module will require user commands sent from the web interface to operate the camera, as well as automatic shuttering, and database updating during security mode. Once this is done, it will be regression tested with all sub-modules currently complete.

### Wi-Fi Module:

#### Front End User Application (Tier 1)

The main point of entrance for all devices is a website hosted on a local intranet. This form of device connectivity was mainly picked for the use of possible video streaming and scalability of functionality to other smart homes and future devices. The full stack framework that will be used is WISA, which stands for Windows, IIS, SQL, and ASP.net.

Below shows the hierarchy of the Graphical User interface



**Phase 1:** The first step will be to set up web interface front end and deploy on IIS. This will be tested by bringing up the web interface on a web browser and verifying that the GUI is operational by testing buttons and other functions.

**Phase 2:** The next step will be to set up and test the connection to the database. Display Data on user interface and change data.

**Phase 3:** Send commands to Wi-Fi Module to check for ability to pass data to Wi-Fi and actuate LEDs. Test sending from Wi-Fi to user interface and database.

**Final Phase:** Final integration to pass command to IIS to log data and to send data to Wi-Fi module, and have the module complete command.

#### Web Server (Tier 2)



The web server will be an application hosted on Microsoft IIS Express, will be written in C#, and will have a backend database using Microsoft SQL Server Express. The web server will be used to interface user commands, device output, and device configurations to the MCU and the Wi-Fi module. The application will communicate with the database to store and load commands. Additionally, it will also log the device history for the user to view.

The PIC32 will communicate to the web application through the Wi-Fi module using the TCP stack. The MCU will communicate device status, send images, sensor data, and be configurable via Wi-Fi module.

The web server will have a total of 3 phases of tests that it will go through during the project’s timeline. The phases are as follows: Phase 1, Phase 2, and Final Phase.

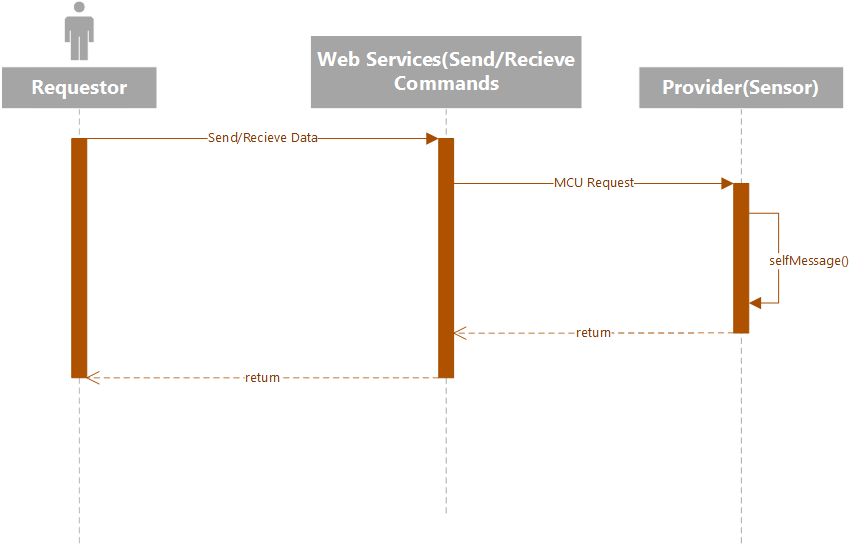
**Phase 1:** For the first phase of testing for the web server, it needs to accept requests from a local machine.

**Phase 2:** During the next phase, different devices will be used to confirm that they all can connect to the web server interface. This is to ensure that the project can be used with a wide array of devices.

**Final Phase:** The final phase will be to test if the web server will accept data pushes. After this is complete, then the interface will be integrated with the rest of the device and use regressive tests for the rest of the parts to ensure there are no additional issues with the module.

#### Web Services Software (Tier 2)

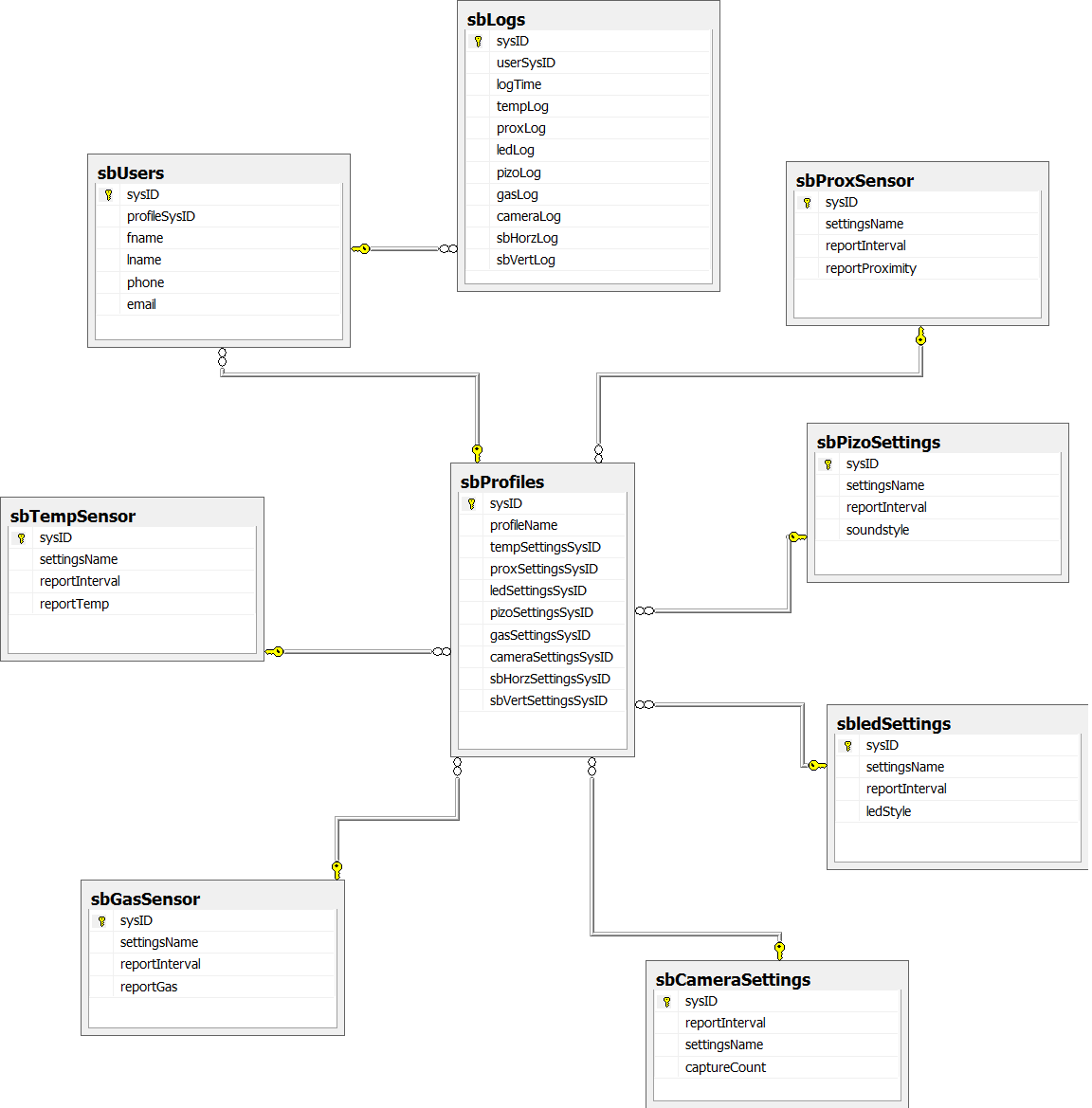
The web services UML flow will operate as below. Requests are sent and routed according to the correct provider for the service.



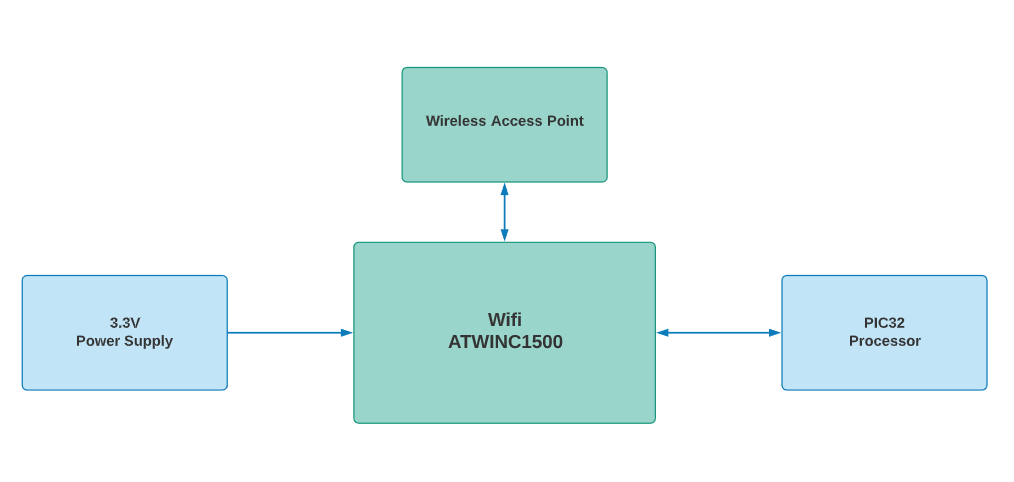
#### Database Interface (Tier 3)

All database functions will be handled by Microsoft SQL Server. The server exposes views, and stored procedures to the client applications and maintains data integrity.

The database will be organized according to the Crow’s Chart below.

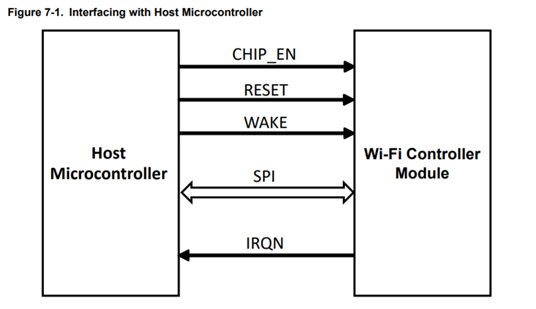


MCU Wi-Fi Hardware (tier 4)



The high-level diagram above shows the relationship between the Wi-Fi module to the PIC32MZ2048EFG144 MCU and the Wi-Fi module to the Software solution. The Wireless Access Point will be interfaced with a WLAN router that will handle DNS, DHCP and TCP/IP connections. From the Access Point we then connect the project to a Web Server. A laptop will be used to handle end user communications from the web-based user interface. Thus, completing project wireless functionality.   


Interfacing with the PIC32 will be done via SPI using the PIC32 SPI module. This can be seen in the image below.



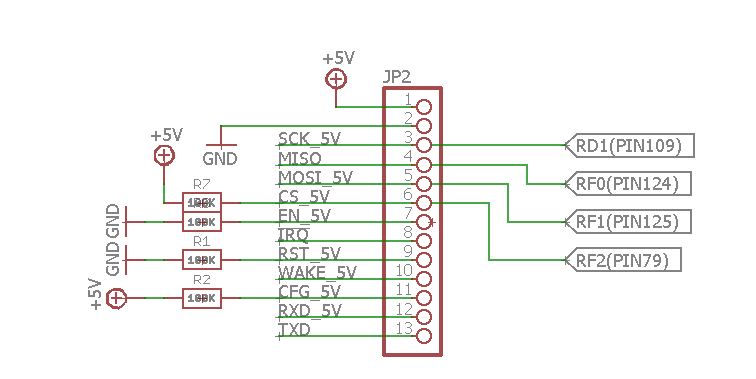
Given a PBCLK of 84Mhz, a SCK of 21 MHz can be achieved by setting the SPI1BRG to 3. This will give a setup time of Tsck/2 - 15 = 23.8 - 15 = 8.8ns which meets the WINC1500’s minimum setup time of 1ns. Likewise, the hold time will be Tsck/2 + 15 = 23.8 + 15 = 38.8ns which meets the WINC1500’s specification of 5ns minimum.

Interrupt services will be written to handle sending and receiving data to the SPI1 module. It will send instructions and data, back and forth between the PIC32 and the web interface. These commands will be sent as an 8-bit opt-code, giving us access to 256 individual commands.

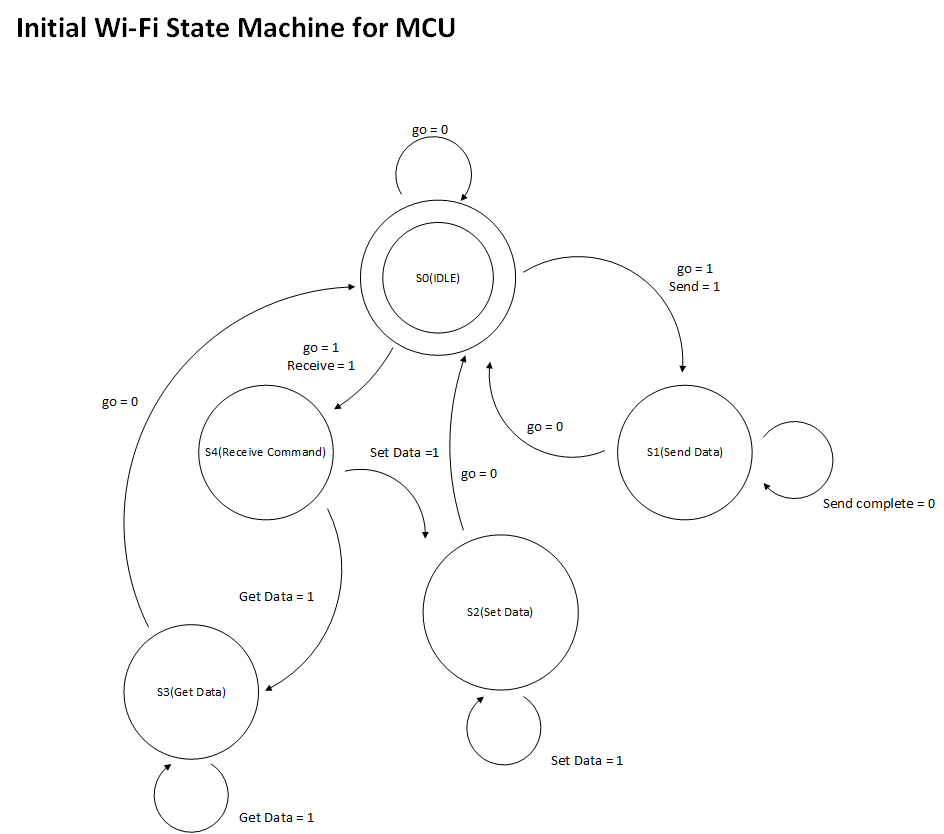
Communication will be sent one character at a time, starting with an opt-code indicating what the following data contains. Below are some examples of opt-codes that will be used, as well as their length and order of data. Given an 8-bit opt-code, this allows us a maximum of 256 possible commands, which is enough for our purposes.

|  |  |  |  |
| --- | --- | --- | --- |
| Command | OPTCODE | Length | Sequence |
| Blinds up | 0b0000.0000 | 1 | cmd |
| Blinds down | 0b0000.0001 | 1 | cmd |
| Open blinds | 0b0000.0010 | 1 | cmd |
| Close Blinds | 0b0000.0011 | 1 | cmd |
| Set auto-shut temp level | 0b0000.0100 | 2 | cmd, new\_temp |
| Log Sensor Data | 0b0000.0101 | 4 | cmd, temp, prox, gas |
| Send images | 0b0000.0110 | n | cmd, img1, img2, ...., imgn |
| etc... | 0b0000.0111 | n+1 | cmd, .... |

The PCB containing the WINC1500 and antenna will be connected to the PIC32 according to the following wiring schematic.



The communication hardware requirements of the project will be met using the Atmel® SmartConnect ATWINC1500 is an IEEE® 802.11 b/g/n network controller SoC for applications in the Internet-Of-Things. Table XXX is a shortened list of features that are relevant to implementation. The Wi-Fi module will communicate device information and data settings to a Web Server. Which will in turn make that information and communication possible with the Automated Smart Blinds MCU, and peripheral devices.



State 0 (IDLE) – Determine if this is a request to Send or Receive data

State 1 (Send Data) – Remain in state until function request is completed. Then, return to IDLE.

State 2 (Set Data) – Determine if received data is requested to get or set data.

* Get Data in form of opt-code and decipher,
* Format data to conform to opt-code and send to MCU.

State 3 (Get Data) - Handle users or systems request to retrieve information from a sensor or MCU.

State 4 (Receive Command) - Receive command and decide which state to send request.

Software Requirements:

The Moore state machine diagram shown above demonstrates the 5 states the software the MCU will need to implement. When called from the main RTS loop, the software checks global variables to actuate a response from the MCU. The state table shown below are the initial values and state requirements being implemented. Variable description is explained below.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Current State | | Next State | | | | | | | | | | | | Output |
|  |  | (AV) Action Variable | | | (SI) Sample Input | | | (ME) Motor Event | | (CE) Command Event | | | | Focus Status |
|  |  | 00 | 01 | 10 | 00 | 01 | 10 | 0 | 1 | 000 | 001 | 010 | 011 |  |
| S0 | 00001 | S0 | S1 | S4 | xx | xx | xx | xx | xx | xx | xx | xx | xx | 000 |
| S1 | 00010 | xx | xx | S4 | S1 | S1 | S2 | xx | xx | xx | xx | xx | xx | 001 |
| S2 | 00100 | xx | xx | S4 | xx | xx | xx | S2 | S0 | xx | xx | xx | xx | 010 |
| S3 | 01000 | xx | xx | S4 | xx | xx | xx | xx | xx | xx | xx | S3 | S0 | 011 |
| S4 | 10000 | xx | xx | xx | xx | xx | xx | xx | xx | S4 | S3 | xx | xx | 100 |

The web interface is hosted on Microsoft IIS 7, and uses Microsoft SignalR for real-time applications and updates. It also uses Microsoft SQL 2008 to handle database related functionality.

The Wi-Fi device will have a total of 4 phases of tests that it will go through during the project’s timeline. The phases are as follows: Phase 1, Phase 2, Phase 3, and Final Phase.

**Phase 1:** The Wi-Fi module will be tested out of the box to see if the Atmel Studio IDE can read the chip, and chip identification number.

**Phase 2:** The Wi-Fi module will be tested against known libraries and a known working MCU that the documentation suggests. In this phase the basic module operations and functions will be learned and tested by going through the training laid out in the documentation.

**Phase 3:** For the next phase, the test will consist of wiring the Wi-Fi module up to the PIC32 MCU. Once this is complete, the test will then be to write and test code for the PIC32, and test communication with the Front-End application.

**Final Phase:** In this phase, the set of tests will consist of communication with other sub-modules to communicate with the Front-End application and actuate device function via the web-based application. It will go through the game regression style of testing with the rest of the sub-modules completed.

# Parts List

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Module | Sub-Module | Choice 1 | Choice 2 | Choice 1 Has in hand (X) | Choice 2 Has in hand (X) |
| Microcontroller |  | PIC32MZ2048ECG144 on Olimex Dev Board | Curiosity PIC32MX470 Development Board | X |  |
| Input Sensors | Proximity Sensor | Sharp GP2Y0A710YK0F Long Range IR Distance Sensor | HRLV-MaxSonar-EZ0 Range Finder | X |  |
|  | Temperature Sensor | TMP36 Analog Temperature Sensor | DS18B20 Digital temperature sensor | X |  |
|  | Smoke/Gas Sensor | MQ135 MQ-135 Air Quality Sensor | MQ-4 Methane/Natural Gas Sensor | X |  |
| Output | LED | Domed SMT5050 RGB LED with custom driver circuit | 7 x WS2812 5050 RGB LED with Integrated Drivers | X |  |
|  | Up/Down Motors | Small Reduction Stepper Motor - 5VDC 32-Step 1/16 Gearing | STEPPERONLINE Stepper Motor Nema 17 | X |  |
|  | Open/Close Motors | Small Reduction Stepper Motor - 5VDC 32-Step 1/16 Gearing | STEPPERONLINE Stepper Motor Nema 17 | X |  |
|  | Speaker | PS1240 Piezo Buzzer | PKM22EPPH2001-B0 Audio Sounder | X |  |
| Communication | Camera | Arducam Mini Module Camera Shield with OV2640 | UCAM-III | X |  |
|  | Wi-Fi Module | ATWINC1500-MR210PB1952 802.11 b/g/n SmartConnect IoT Module | AC164164 - PIC-IoT WG | X |  |
|  | Wi-Fi Module Supplemental | Atmel SAM D21 Xplained Pro Evaluation Kit |  | X |  |
| Power Supply | AC to DC Stepper Supply | RS-15-5 5V 3A Switching Power Supply | 12Vdc 5A Power Supply Screw Term Blocks | X |  |
| Blinds |  | Micro metal blinds |  | X |  |

# Preliminary Cost

The estimated cost for this device is $19,529.27. For the first section of the overall costs: the parts are calculated by the primary set of parts ordered for the project. The cost for the parts may vary if any of the primary parts are not able to be implemented into the project.

For the second section, the engineering cost is based off the average pay of an Embedded Systems Engineer, divided by 2 since the crew would be internship levels of pay. This also incorporates on how many weeks with how many hours of work per week when calculating the price.

For the final section being the Outsourced contract cost, the initial price of $0 will be since this is a junior project done by students. If necessary, then the price for having an outsourced contract will go up to a maximum amount of $5000.

Some of the errors for this cost can include time required to finish development, more or different parts needed for the project, and the possibility of having to outsource a contract to a third party.

|  |  |
| --- | --- |
| Estimated total cost | $19,529.27 |
| Estimated parts/packaging cost | $259.27 |
| Engineering/labor cost | $19,200 |
| Outsourced contract cost | $0, if needed $5000 |

# Team Assignment

Chad - In charge of hardware implementation, and the various input/output parts. Master document proofreader.

Drew - Overall responsibility of project. In charge of electrical circuits, Microprocessor module integration and documentation.

Michael - In charge of software implementation of the web server, Wi-Fi modem, app software, and the communications between all software.

# References

Links to all current documentation and code are below:

Meeting Notes: [https://docs.google.com/document/d/1mtUfFTfOqqPw2zyaepzclxoFKLoXaqd9-opKlfBveFA/edit#](https://docs.google.com/document/d/1mtUfFTfOqqPw2zyaepzclxoFKLoXaqd9-opKlfBveFA/edit)

Control Documentation:

[https://docs.google.com/document/d/1RbYqEY9eu72LCQb83p9FujjQguTNRpgIjmdwpXFczCA/edit#](https://docs.google.com/document/d/1RbYqEY9eu72LCQb83p9FujjQguTNRpgIjmdwpXFczCA/edit)

Plan Documentation:

[https://docs.google.com/document/d/1JT-my9udzIbytgEOQ\_zhH8BzzEN3PZicPn5nsdUEtqM/edit#](https://docs.google.com/document/d/1JT-my9udzIbytgEOQ_zhH8BzzEN3PZicPn5nsdUEtqM/edit)

Schedule Documentation:

<https://docs.google.com/document/d/10Dt15eweoR7eboyIdi2v3zc0MMXtXyCiVJ5iHuWIVT8/edit>

GitHub Repository (includes Gantt Chart, datasheets and all code):

<https://github.com/TheJonBovi/AutomatedSmartBlinds>