Vehicle Interchangeable Electronic Controller Network System

K-RAM

Version 1.0

5/7/2018

Final Report

Version 1.0

Contents

A. Introduction

1. Purpose

Whenever a space shuttle goes into space, a number of controllers are needed to control the various systems in the craft. If a controller breaks, it will need to be replaced. One option is to send replacements, but for a shuttle far way from earth, this would be too time consuming. Taking extras of each type of controller is also an impractical option because they add to the weight of the cargo, which translates into a greater cost to get enough fuel to send it all into space. The Vehicle Interchangeable Electronic Controller(VIEC) Network System aims to alleviate some of this burden by creating a system of interchangeable controllers (IC) that can be interchanged into the network at any time. The server that manages this system loads the appropriate application IC whenever a new one has been plugged in. A universal connector will be used to interface the ICs with the corresponding input devices. The simulated systems will be a Habitat Lighting System, Environment Monitoring System, and a Reaction Control System.

2. Functional Requirements

ID	Functional Requirements	Team Member Responsible	Effort (in %)
1	The IC shall display output and status updates to an LCD.	Kaothar	25
2	The ICs shall run Application 1, 2, or 3, which are stored on the server.	Richard	20
3	The server shall manage the storage of Applications.	Richard	35
4	The server shall manage the distribution of Applications.	Richard	35
5	The server shall detect the status of each IC.	Richard	10
6	IC 1 shall have Application 1 which would be able to get an input from a sensor and produce some output.	Kaothar	40
7	IC 2 shall have Application 2 which would be able to get an input from a sensor and produce some output.	Alberto	40
8	IC 3 shall have Application 3 which would be able to get an input from a sensor and produce some output.	Marco	40
9	The server shall have a GUI that shows the status of the ICs	Marco	45
10	The server shall display instructions to the user on how to replace the IC	Marco	10

3. Non-Functional Requirements

ID	Non-Functional Requirements	Team Member Responsible	Effort (in %)
1	The IC shall be no larger than 150mm × 100mm × 50mm	Kaothar	20
2	The IC shall require a power supply of at least 5V and 2A.	Kaothar	5
3	There shall be a total of 3 output devices.	Kaothar	10
4	The IC shall have a “universal” connector to interface with the network.	Alberto	30
5	The server shall be no larger than 250mm x 130mm x 90mm	Marco	5
6	The ICs shall be able to swap between different applications	Alberto	15
7	There shall be a total of at least 5 sensors.	Alberto	15

4. Constraints

a. The applications loaded on each controller cannot be too large because the server will need to have a backup of each application plus the code for itself to run. Otherwise, the server would not be able to store everything.

b. The server shall not take an excessive amount of time to recognize a fault in the system.

c. The server shall not take an excessive amount of time to reconfigure the controllers.

d. Each application shall use at least one input and output.

e. The system shall have one server.

f. The system shall have three ICs.

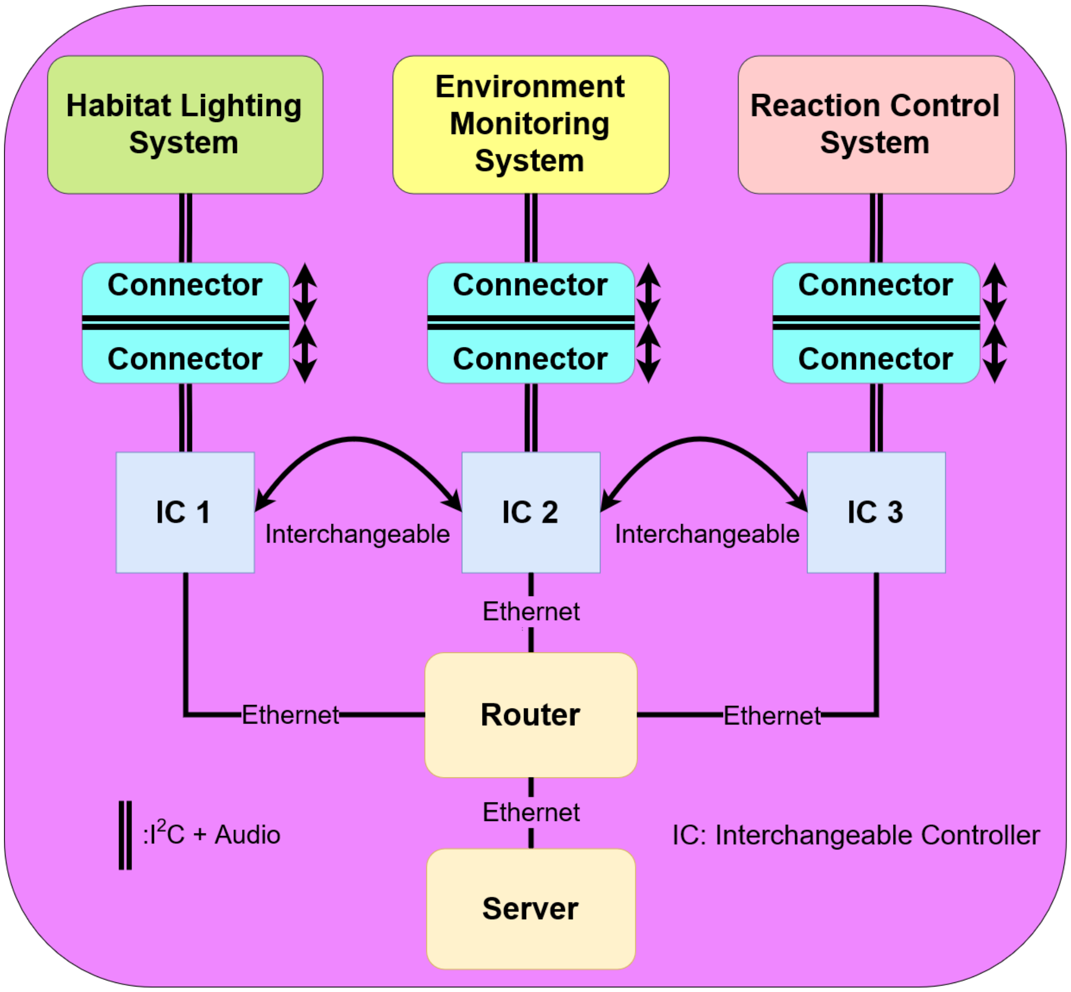
5. System Overview

a. System Overview

The VIEC Network System is a system comprising of a server, three interchangeable controllers, and three applications. The server manages the transfer of applications to each interchangeable controller (IC) as well as keeping track of their status. Each controller is tasked with running a unique application. The applications are a Habitat Lighting Controller, Environment Monitoring Controller, and Reaction Control System. Whenever an IC fails, it can be replaced by any of the other ICs provided they are not in use. Once the IC has been replaced, the server will reconfigure the IC so the application can restart.

b. System Organization

In this system, the server and interchangeable controllers (IC) are connected to a router over Ethernet which facilitates communication between them. The ICs are also connected to each application subsystem via a universal connector.



B. Sub-System Server

1. Sub-System Server Overview

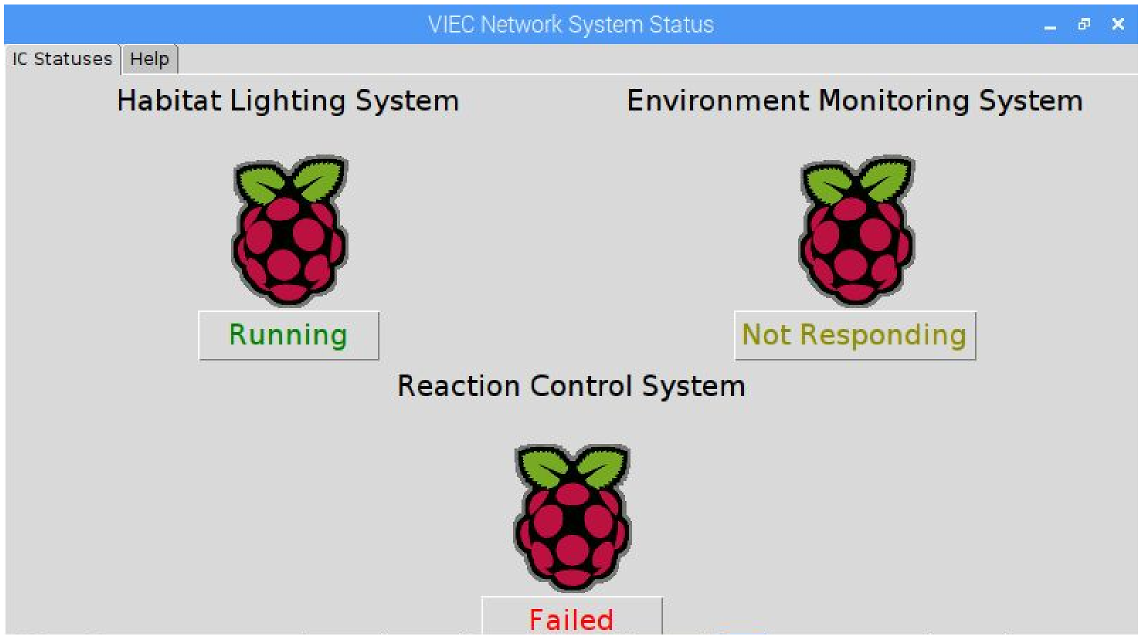
The server is center of the VIEC system. It distributes applications to the IC via the router. It then monitors the ICs for any faults in its application and displays each IC's current status to a touchscreen display. When a fault is detected, the server also redistributes the IC's corresponding application to the replacement IC once it has been plugged into the network.

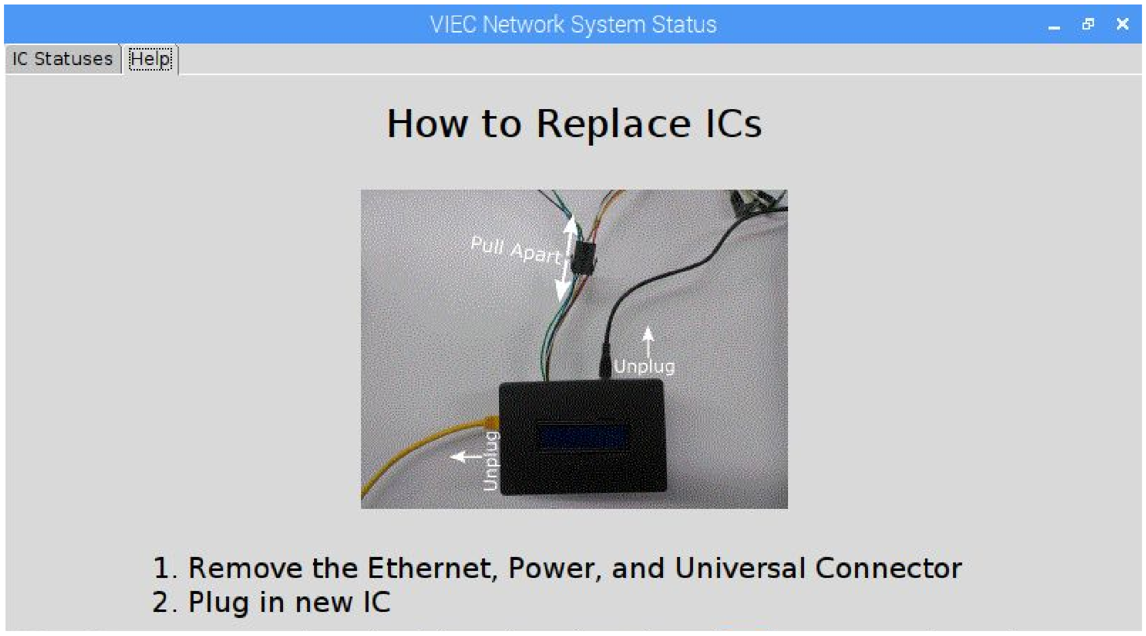
2. Sub-System Server Organization

The server.py script is the core of the serer program. It starts by spawning two processes, One runs the graphical user interface (GUI) and the other remotely shuts down ICs. Then the server-side of the SSL Socket is created. The connection process begins next and a list of connected ICs is generated. The server sends each IC script needed to control the system it is connected to. The IC then begins running the files delivered and connects back to the server with the client-side of an SSL Socket. This socket connection is used to inform the server of the status of the application. The server monitors the messages sent and updates the database whenever an IC's status changes. The following subsections describe in greater detail how the aforementioned functionality is accomplished.

a. Displaying the Status

The server physically consists of a Raspberry Pi, a 7" LCD Touchscreen Display, and a case for the display. The server interfaces with the Touchscreen via a DSI Ribbon Cable that comes with the Display. The LCD is used to display the GUI. The GUI informs the user of the current status of each IC and can remotely shutdown any connected IC. The GUI can also be used to shutdown the entire system by clicking the 'X' at the top right of the screen. The first image below is the IC Status tab of the GUI. Also, a help screen can be shown at any time to describe how to replace an IC. The second image shown below is the Help Tab of the GUI. The code used to create the GUI is serverGUI.py.





How to Replace ICs

1. Remove the Ethernet, Power, and Universal Connector
2. Plug in new IC

b. Connecting to Each IC

The server is connected to a router using an ethernet cord. The server uses this connection to detect connected ICs and monitor known ICs' statuses. The code used to detect new ICs and add them to the database is connection.py. Whenever a new IC is connected, the server sends position.py to the IC and then runs it. This script outputs the hardware identification number that corresponds with the system that IC is connected to. Afterwards, the server sends runApplication.py and the application file that corresponds with the hardware identification number received, e.g. App1.py the hardware identification number equals 1.

c. Managing the Database

The database stores the IP address of connected ICs, the hardware identification number for the application that they're running and the current status. The statuses that can be stored are RUNNING (the application is functioning correctly), CAUTION (the application hasn't responded for 15 seconds), and FAILED (the application has stopped working). An application that doesn't have an IC connected to is shown as offline. The applications are signified by the numbers one through three with one representing the Habitat Lighting System, two representing the Environment Monitoring System, and three representing the Reaction Control System. The code used to facilitate the connection, retrieval, and updates to the database is IC_Database.py.

d. Running the Applications

When the runApplication.py script starts, it performs two tasks. It creates an client-side SSL Socket connection to the server and starts the application, e.g. the App1.py file. As the application runs, it prints various messages stating what is currently happening. The runApplication.py script analyzes these messages to determine if it is a regular status message or an error message. The result of this evaluation is sent through the socket to inform the server whether the system is functioning correctly or if it has failed.

3. Components

a. 1 Raspberry Pi 3 Model B

b. 1 LCD Touchscreen with Case

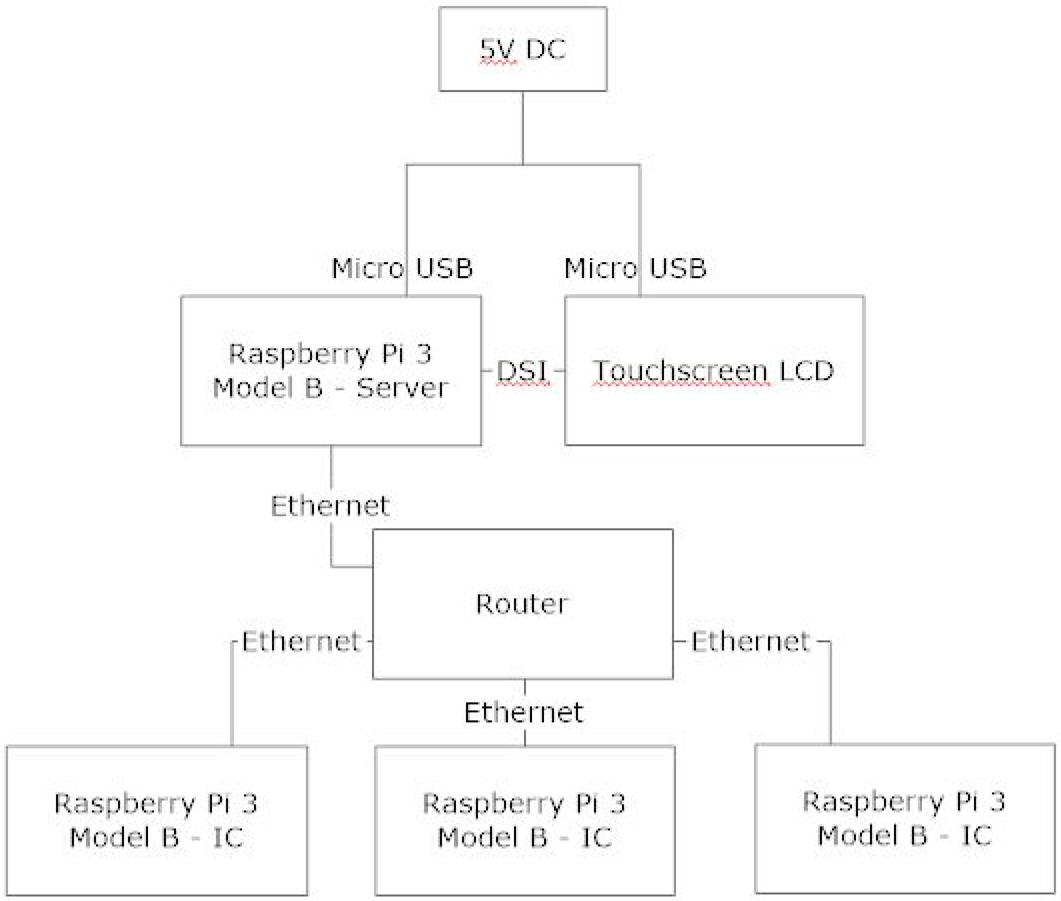
c. 1 Router

d. 4 Ethernet Cables

e. 1 MicroSD Card

f. 1 5V Power Supply

4. Schematic


5V DC
Micro USB
Micro USB
Raspberry Pi 3 Model B - Server
DSI
Touchscreen LCD
Ethernet
Router
Ethernet
Ethernet
Ethernet
Raspberry Pi 3 Model B - IC
Raspberry Pi 3 Model B - IC
Raspberry Pi 3 Model B - IC

5. Software Code

a. Written Code

All code is stored in UNT's server. The files uploaded there are as follows:

- server.py - Python script that provides the main functionality of the server by monitoring connected ICs of status changes and updating the database whenever a change occurs,
- serverGUI.py - Python script that defines the server's GUI and how it should behave
- IC_Database.py - Python script that defines how that server can access the database
- connection.py - Python script to facilitate the connection of new ICs to the Network
- position.py - Python script to determine what the hardware identification number is for the system that IC is connected to
- runApplication.py - Python script that runs each application on the IC, logs all messages, and informs the server of the status of the application.
- App1.py - Python script that controls the Habitat Lighting System

- App2.py - Python script that controls the Environmental Monitoring System
- App3.py - Python script that controls the Reaction Control System

b. Third Party Software Needed

- Raspbian - https://www.raspberrypi.org/downloads/raspbian/
- Python version 3.5 - https://www.python.org/
- tkinter - https://wiki.python.org/moin/TkInter
- Pillow - http://python-pillow.org/
- Paramiko - http://www.paramiko.org/
- SQLite3 - https://github.com/python/cpython/tree/3.5/Lib/sqlite3

C. Sub-System Interchangeable Controller

1. Sub-System Interchangeable Controller Overview

The interchangeable controller is a piece of hardware that is able to interface with all of the application subsystems. When the controller is plugged into one of the subsystems it reads the ID of the system and reports it back to the server which replies with the corresponding application to be run.

2. Sub-System Interchangeable Controller Organization

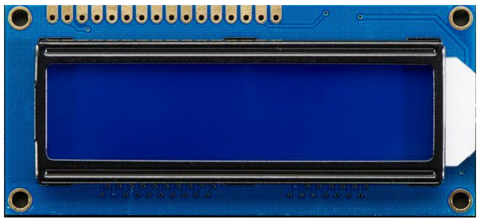
The interchangeable controller is implemented using a Raspberry Pi, an LCD to display data, and a universal connector which will be covered in the next section.

3. Components

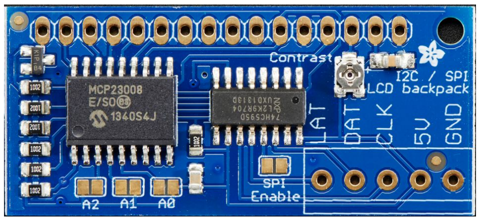
a. 1 Raspberry Pi 3 Model B

b. 1 1591XXTS Hammond Enclosure

c. 1 16x2 HD44780 LCD



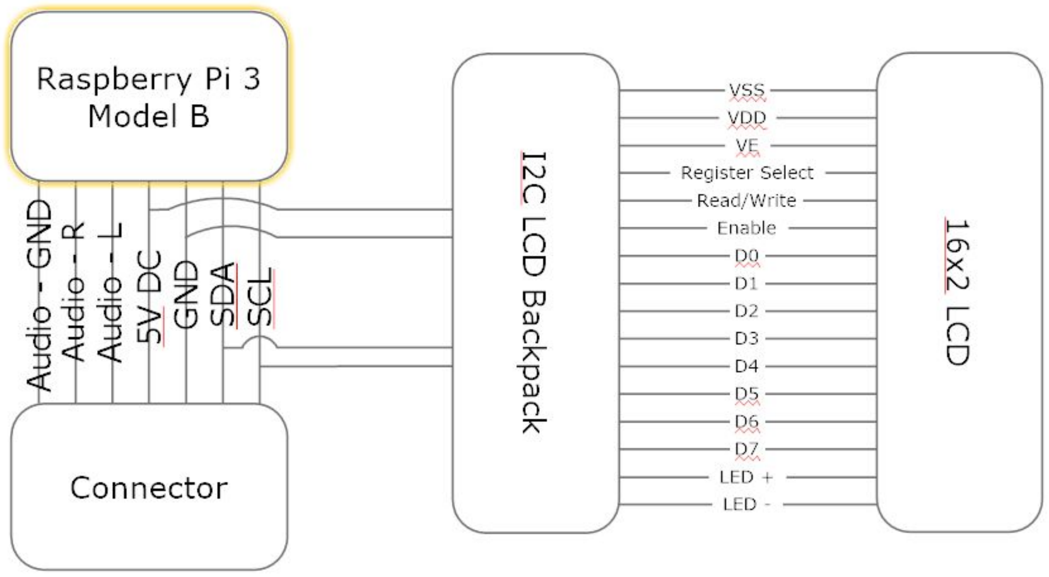
d. 1 Adafruit i2c / SPI character LCD backpack



e. 1 MicroSD Card

f. 1 5V Power Supply

4. Schematic



5. Software Code

a. Written Code

All code is stored in UNT's server. The files there are as follows:

- App1.py - Python script that controls the Habitat Lighting System
- App2.py - Python script that controls the Environmental Monitoring System
- App3.py - Python script that controls the Reaction Control System

b. Third Party Software Needed

- HTU21DF Library - https://github.com/dalexgray/RaspberryPI_HTU21DF
- LCD Display Library - https://github.com/adafruit/Adafruit_Python_CharLCD
- LED Matrix Library - https://github.com/adafruit/Adafruit_Python_LED_Backpack
- Analog to Digital Library- https://github.com/adafruit/Adafruit_Python_ADS1x15

- GPIO Library - https://github.com/adafruit/Adafruit_Python_GPIO
- PWM Library - https://github.com/adafruit/Adafruit_Python_PCA9685
- GPIO Expander Library - https://github.com/adafruit/Adafruit_CircuitPython_MCP230xx

D. Sub-System Universal Connector

1. Sub-System Universal Connector Overview

The universal connector will be able to connect each of the Interchangeable Controllers (ICs) to all of the application subsystems. With the universal connector the IC would be able to communicate with the sensors and audio if needed in order to run each application properly.

2. Sub-System Universal Connector Organization

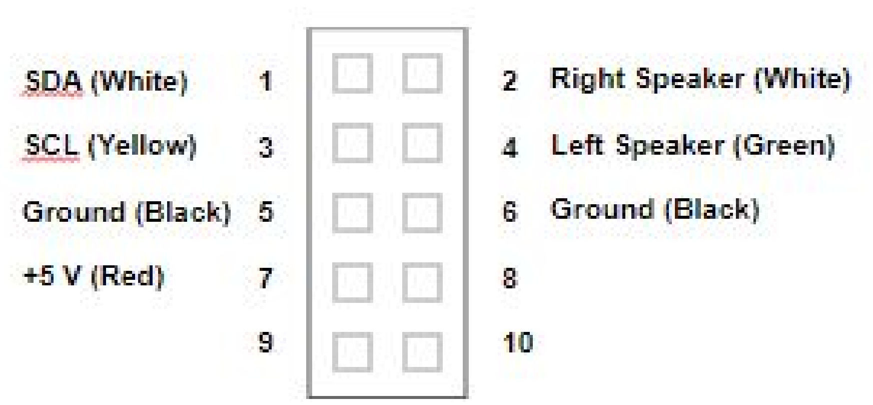
The Universal Controller is implemented by using a Male 43025-1000 Molex Connector that will be integrated with the IC and a Female 43020-1000 Molex Connector that will be integrated with each application system. The connector will carry the I^2C serial bus needed to run some of the applications and audio. The system uses 3 universal connector pairs.

3. Components

A. 3 Female 43020-1000 Molex Connector

B. 3 Male 43025-1000 Molex Connector

4. Schematic



5. Software Code

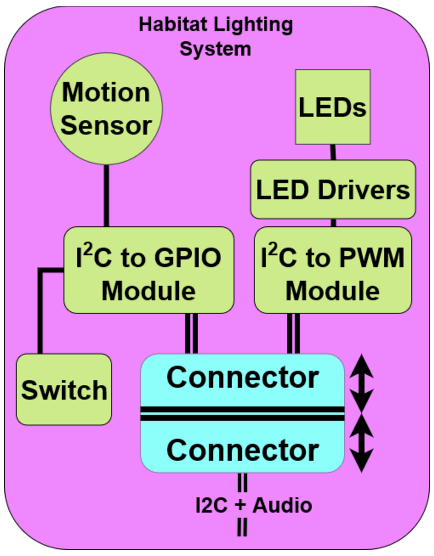
Not applicable

E. Sub-System Habitat Lighting System

1. Sub-System Habitat Lighting System Overview

This application increases the brightness of two LED lights to 100% whenever motion is sensed and reduces their brightness to 10% when motion is no longer detected.

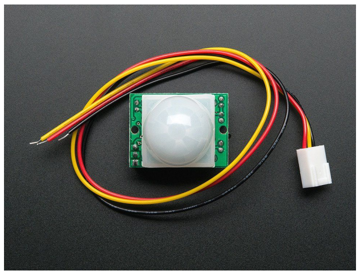
2. Sub-System Habitat Lighting System Organization



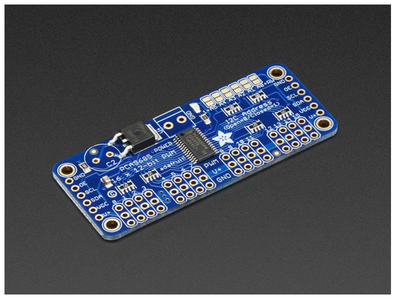
In this system, the interchangeable controllers (IC) and the Habitat Lighting system are connected over the Universal Connector which carries both the I²C serial bus and audio. The system contains a I²C to PWM module to connect to the LEDs and a I²C to GPIO to connect to the switch and the motion sensor.

3. Components

a. 1 PIR Sensor



b. 1 PCA9685 PWM Driver

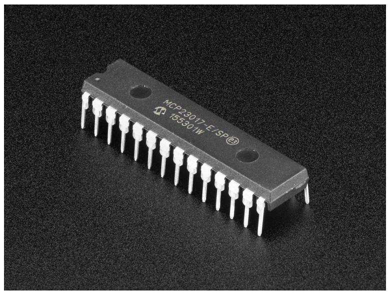


c. 1 Power Splitter Cable

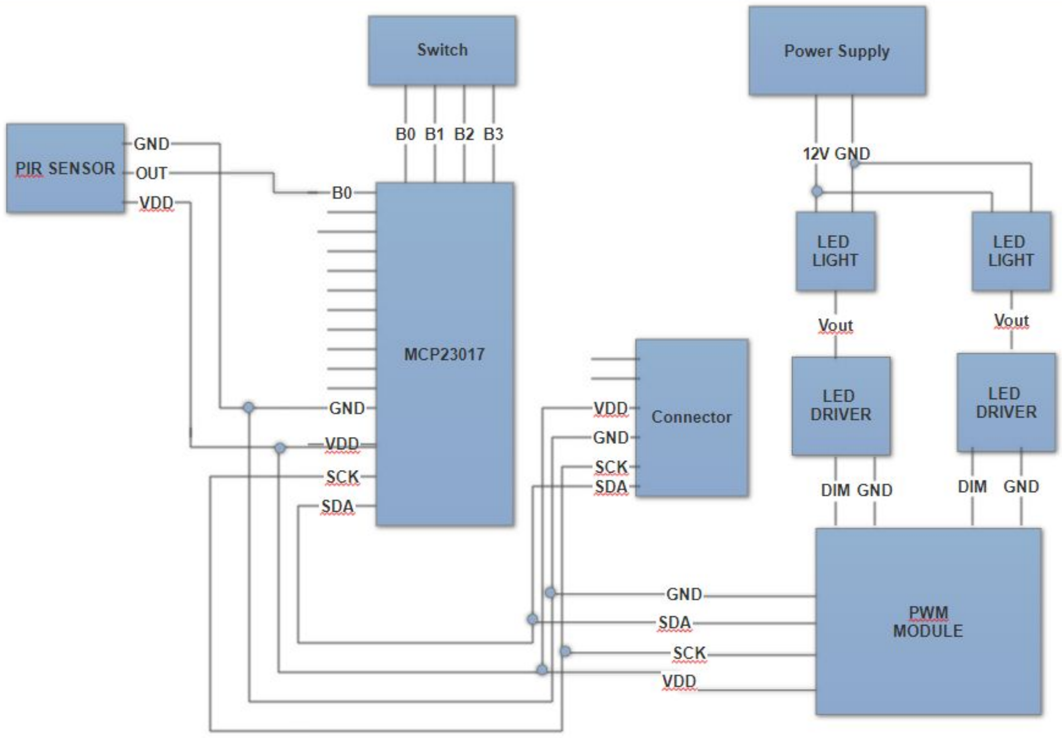
d. 2 LDD-1500LW Mean Well LED Drivers

e. 2 LED Puck Lights

f. 1 MCP23017 I2C I/O expander



4. Schematic



5. Software Code

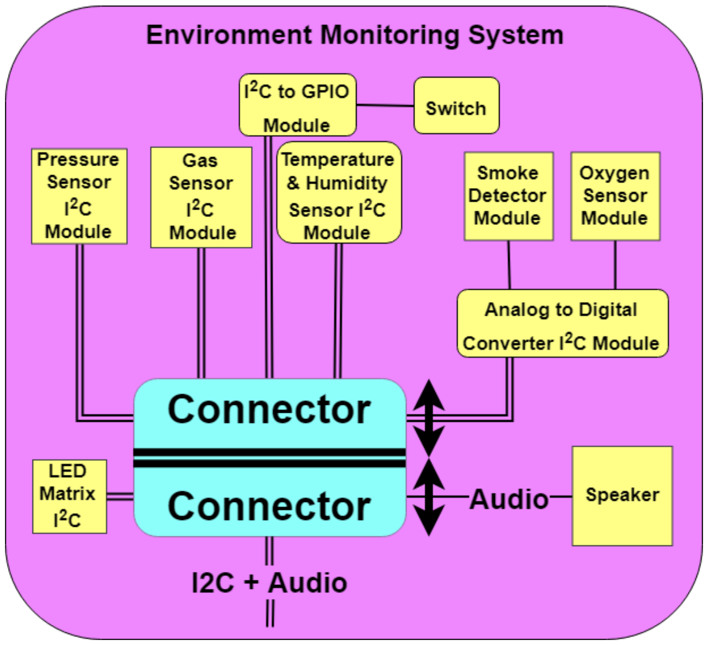
Not Applicable.

F. Sub-System Environment Monitoring System

1. Sub-System Environment Monitoring System Overview

This controller shall monitor various environmental levels. Whenever any of the readings reach an unsafe level an alert is displayed on the ICs LCD screen and an alarm is played through a speaker.

2. Sub-System Environment Monitoring System Organization



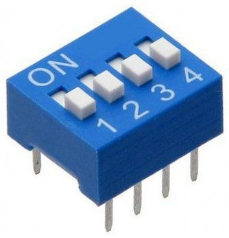
In this system, the interchangeable controllers (IC) and the Environment Monitoring System are connected over the Universal Connector which carries both the I²C serial bus and audio. The system contains a Analog to Digital converter in order to communicate with the Smoke Detector and the Oxygen Sensor, a I²C to GPIO to connect to the switch, a speaker to output audio, and a Pressure, Gas, Temperature, Humidity, and LED matrix which all use the I²C bus.

3. Components

a. 1 Male 43025-1000 Molex Connector

b. 1 1591XXTS Hammond Enclosure

c. 1 DIP Switch

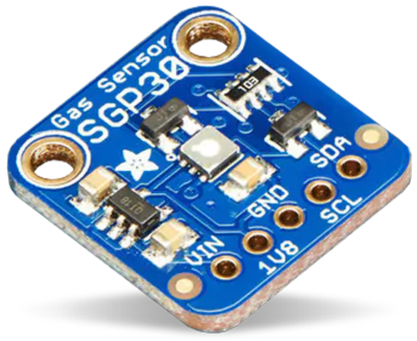


d. 1 Speaker

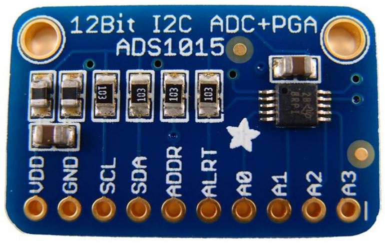
e. 1 LED Matrix

f. 1 MPL3115A2 Pressure Sensor

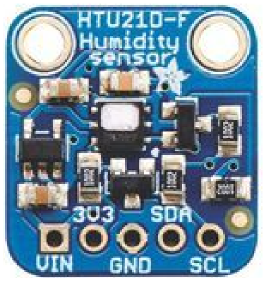
g. 1 SGP30 Gas Sensor



h. 1 Analog to Digital Converter

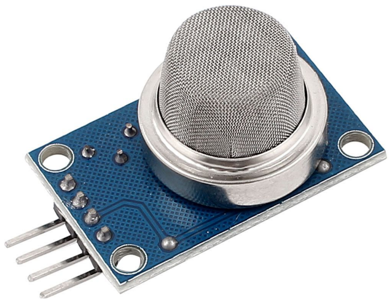


i. 1 Temperature & Humidity Sensor

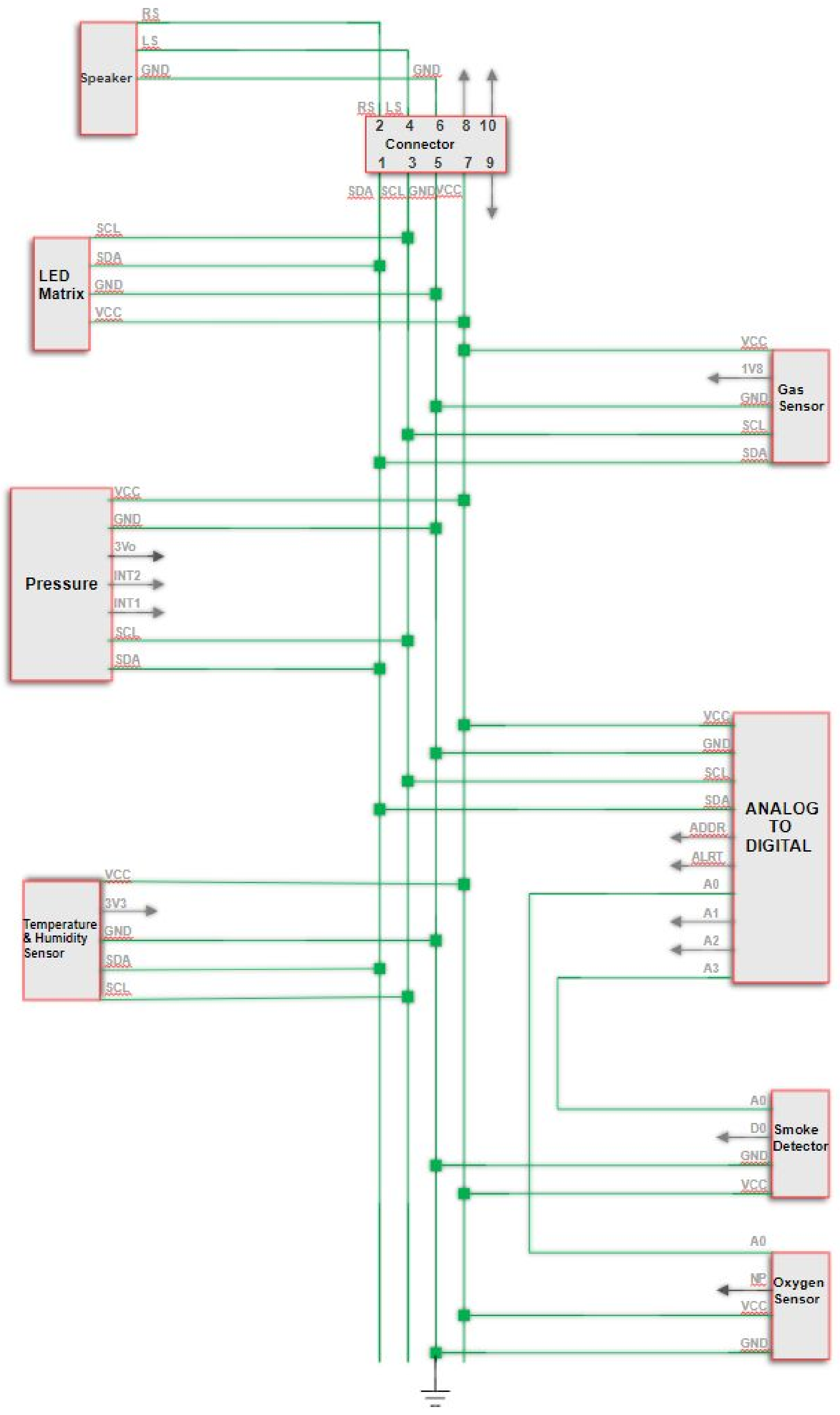


j. 1 Oxygen Sensor

k. 1 Smoke Detector



4. Schematic



5. Software Code

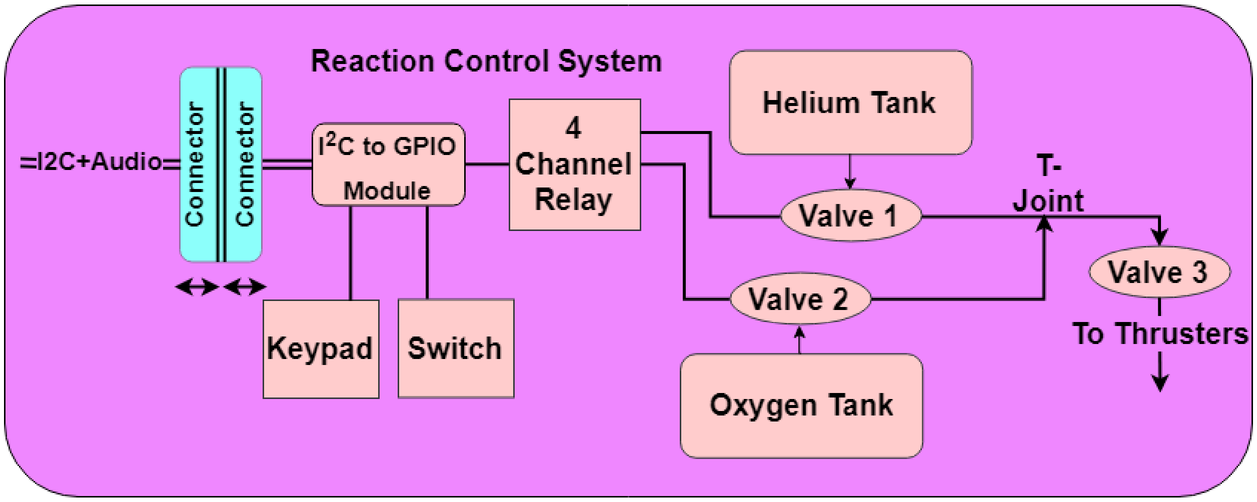
Not Applicable

G. Sub-System Reaction Control System

1. Sub-System Reaction Control System Overview

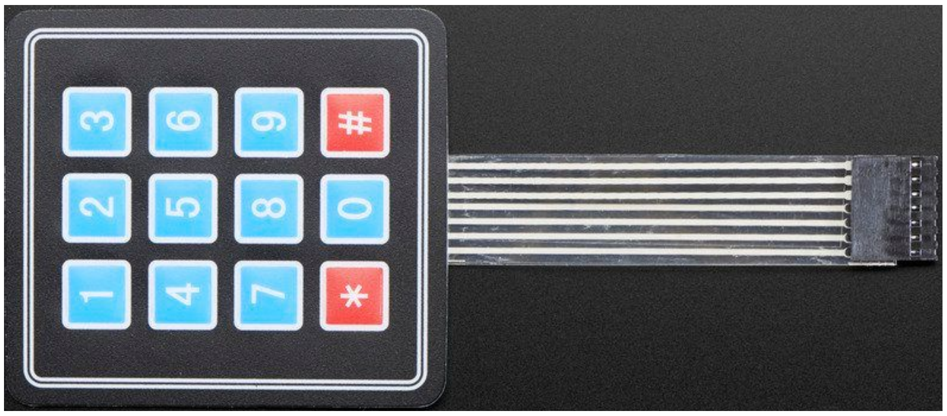
This application will demonstrate a reaction control system with 3 solenoid valves being controlled by the ICs.. Whenever the correct number sequence is entered into the keypad the controller shall open the valves via the relay to allow the contents of the two tanks to mix and then head to the thrusters. In the demonstration the water will be collected in another container and then put back into the helium and oxygen tanks.

2. Sub-System Reaction Control System Organization

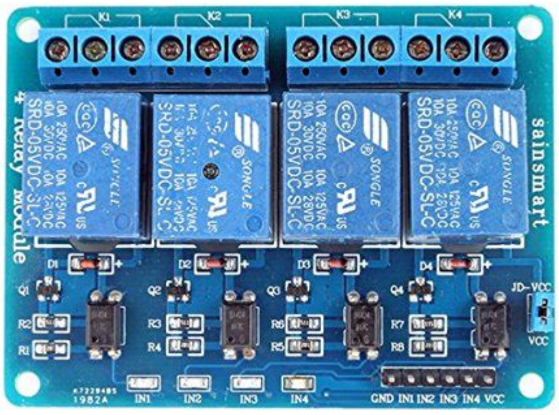


3. Components

a. 1 3x4 Matrix Keypad

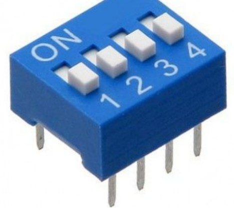


b. 4 channel relay

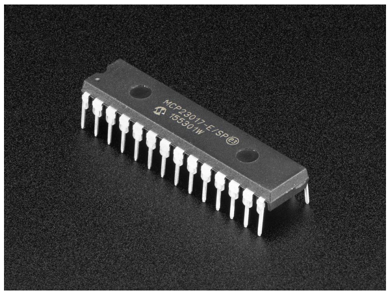


c. IEC320 C14 4P DPST Male Power Socket

d. 1 DIP Switch



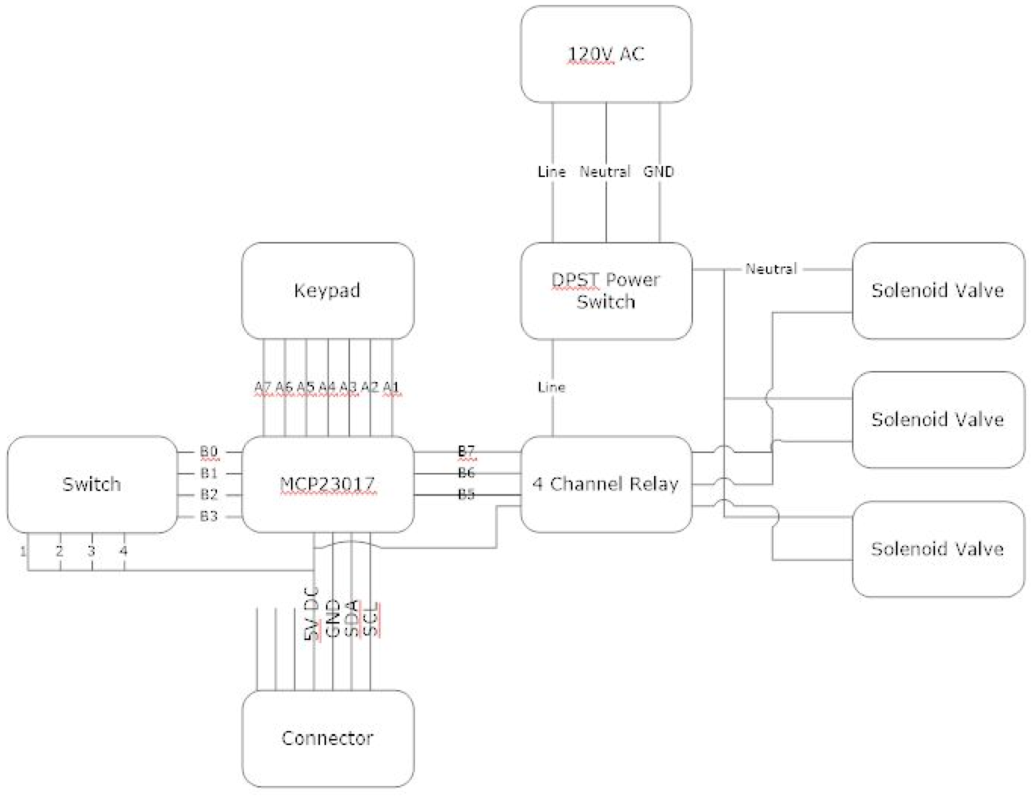
e. 1 MCP23017 I2C I/O expander



f. 3 Solenoid Valves

C. 1 Male 43025-1000 Molex Connector

4. Schematic



5. Software Code

Not Applicable.

H. Results

The team succeeded in all of our endeavours. We created a truly interchangeable controller network. Whenever a controller fails, it can be replaced by any other controller in the network. The server recognizes the newly replaced controller and loads the proper software upon the controller. The following are the swap time statistics.

Activity	**Time (sec)**
Server Recognizes an IC is offline	8.5
Shutting down another IC from the GUI	2.14
Swapping ICs	27.07
Server loads Application on IC and IC begins running	24.4
Total	1 min, 2.11 seconds