Design Document

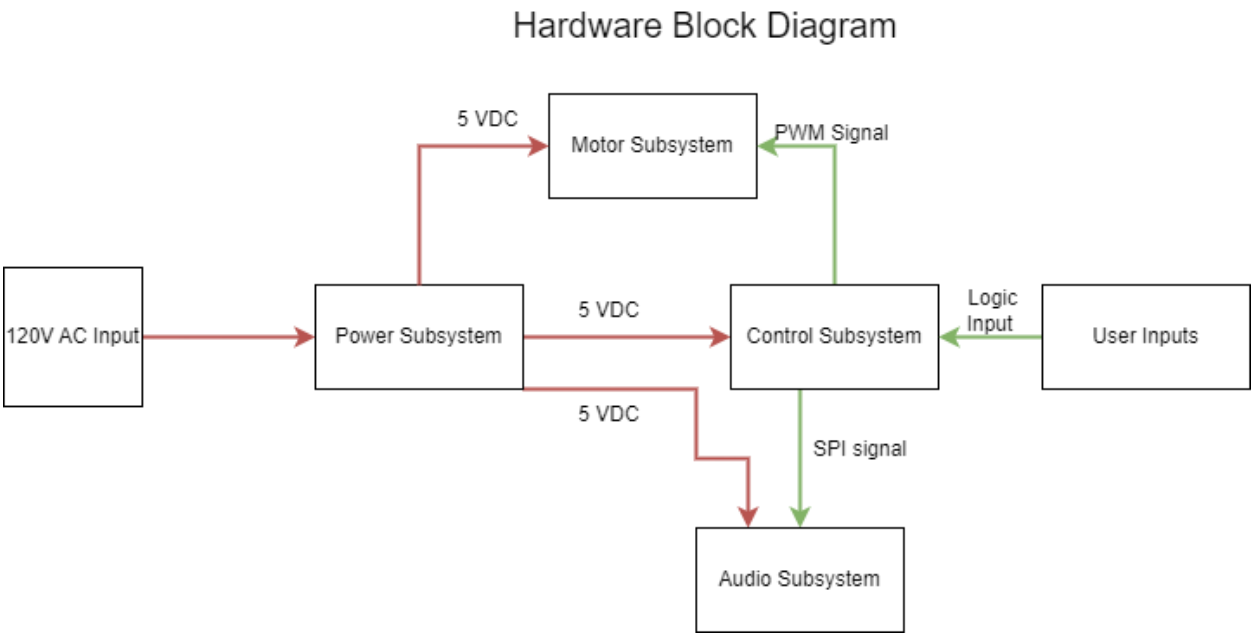
Team B04: Joshua Bristow, Benjamin Farrell, Michel Maalouli, Abdulrahman Tabbaa

Contents

Revision History

Date	Author	Modifications
9/25/2022	Abdulrahman Tabbaa	Made the document. Gathered the Software Architecture, Hierarchical Design, and Interface Description.
10/14/2022	Abdulrahman Tabbaa	Added the Skeletal Framework for later incorporation of each subsystem's design.
10/15/2022	Abdulrahman Tabbaa	Added description for Audio and Power Subsystems. Also modified Interface Description by removing Lighting Subsystem as it does not need to be its own entity separate from Control Subsystem.
10/16/2022	Abdulrahman Tabbaa	Added the Software Hierarchical Design Michel made yesterday.
11/15/2022	Michel Maalouli	Added software architecture description.
11/15/2022	Abdulrahman Tabbaa	Modified Audio input voltages from 4.5 VDC to 5 VDC. Removed any mentions of Director connection button. Removed mentions of diode clamping safety circuitry. Added a Completed Hardware Section

Interface Description



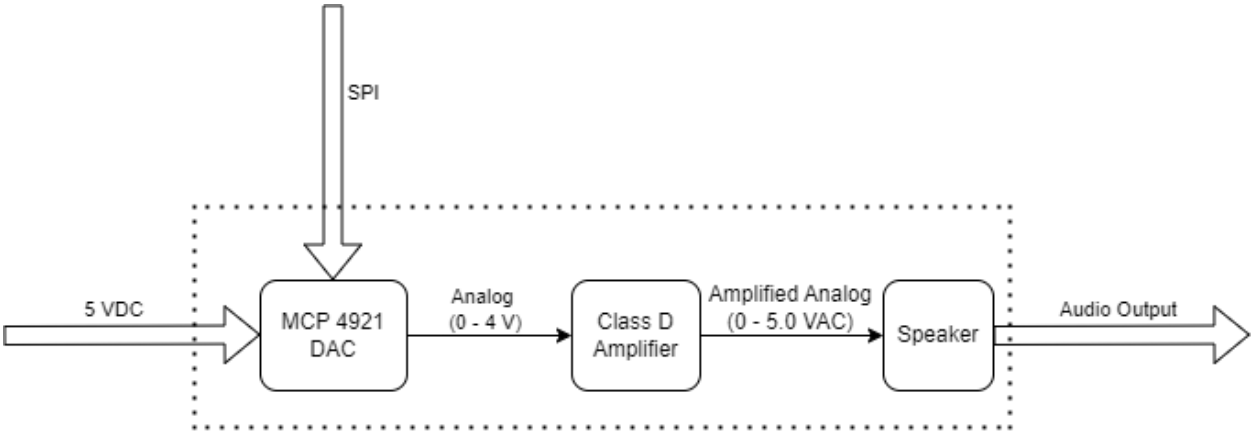
Connection	Source	Destination	Description
1	Power input	Power sub system	120V AC will input in to the power subsystem to be converted and distributed
2	Power subsystem	Controller	5V DC will input into the controller.
3	Power subsystem	Motor subsystem	5V DC will input into the motor sub system
4	Power subsystem	Audio subsystem	5V DC will input into the audio subsystem
5	Controller	Audio subsystem	SPI signal will input to the audio subsystem
6	Controller	Motor subsystem	PWM and/or I2C signal to control the mechanical subsystem
7	User Input	Controller	There will be a test button to individually test audio subsystem, motor subsystem, and lighting (embedded in control subsystem).
8	User input	Controller	There will be a button to have the entire system run the demonstration.

Subsystems

Audio

The purpose of the audio subsystem is to generate audible noise. The main component in this subsystem is the DAC; the technical specifications of the DAC dictate the I/O parameters. For example, the DAC communicates via SPI. Therefore, the audio subsystem takes in SPI from the Control Subsystem as opposed to I2S or PWM. Additionally, this subsystem takes in 5 VDC from the Power Subsystem.

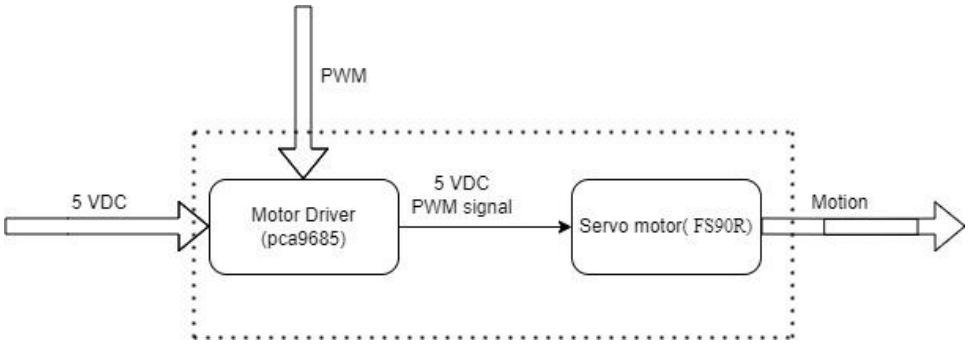
Originally, we considered adding a diode clamping safety circuit to this audio subsystem. However, this would have required pulling the diodes up to a voltage greater than the 5 VDC. Since there was no easy way to do this with the transformer already providing the 5 VDC, this idea was never implemented. Additionally, the transformer contains safety circuitry, so having a diode clamping circuit is not as necessary.



The output of the DAC is not loud enough. As a result, the analog signal gets fed into a Class D Amplifier. The amplified analog signal then gets fed into a speaker which outputs the music/ audio. It was not known when designing whether or not a low-pass filter would be needed. After testing the circuit on a breadboard, the low-pass filter was reducing the noise level significantly. The class D amplifier already contained a low pass filter, so adding an external one was not necessary.

Mechanical

The mechanical system is composed of one servo, the FS90R, to create motion of the hanging clock. This will be done by powering the motor driver, the PCA9685, with 5 VDC. The driver will also receive a PWM signal from the Rasbery pi to control the speed and rotation of the servo motor. The servo will be connected to the swinging clock. The clock will be suspended by a simple hinge on the hand of the rabbit, and the servo will be able to create motion through the hinge.

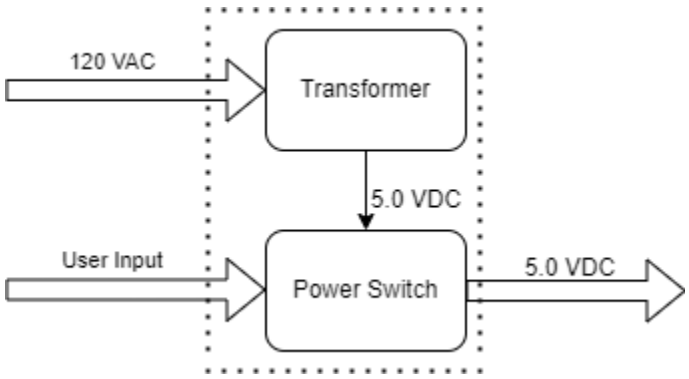


Power

The power subsystem is responsible for converting the power from a 120 VAC wall outlet to different forms usable by the remaining components in the device. The voltage and/or current rating and power consumption of the main components in the system are listed below:

- DAC: 5.5 VDC at 350uA is 0.001925 W
- Class D Amplifier: 5.0 VDC rated at 2.5 W
- Servo Motor (x2): 5.0 VDC at 360 mA is 3.6 W each
- Raspberry Pi: Won't consume more than 4 W

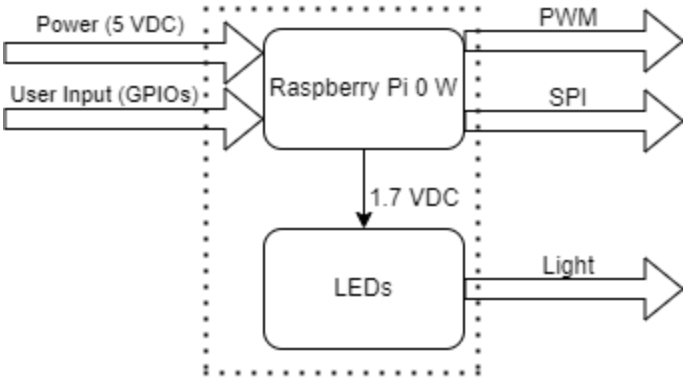
Adding the power constraints from each component, the total power is less than 15 W. With this in mind, and observing that each component is capable of operating on 5 VDC, the appropriate conversion from the power subsystem is 120 VAC to 5 VDC. The motor driver and Pi have safety circuitry, but the DAC and Class D Amplifier, both of which are parts of the Audio Subsystem, do not. There are many ways to implement safety circuitry, but the cheapest yet most effective approach is a diode clamping circuit. Unfortunately, the diodes must be connected to a voltage higher than the voltage input into the Audio Subsystem. Instead of generating a voltage greater than 5.0 VDC to clamp the diodes at, the input to the Audio Subsystem can be simply stepped down to 4.5 VDC. This was not as necessary when it became apparent the transformer contains safety circuitry. Therefore, no step-down of the voltage was implemented.



The conversion from 120 VAC to 5 VDC is performed by a 25 W rated transformer with safety circuitry. The entire device contains a user-accessible power switch. When the switch is closed, the 5 VDC can be propagate to the motor driver and Rasberry Pi.

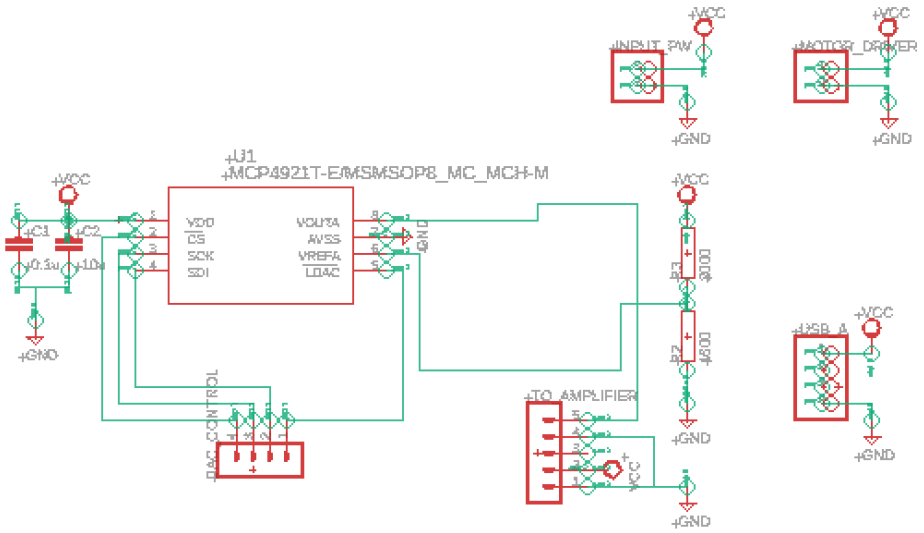
Control

The control subsystem is the "brains" of the entire system. This subsystem includes the Raspberry Pi, which interacts with the other subsystems and facilitates the accurate, sequential performance of events. The control subsystem receives 5 VDC to power on the Pi and user input to dictate which operations (only play audio, only generate motion, only light up, only test Director connection, or perform all four operations) to perform. This subsystem then outputs a PWM signal when controlling motor motion; a SPI signal is generated when audio needs to be played.



Completed Hardware Design

The Audio Subsystem circuit and routing of the 5 VDC from the transformer to the Raspberry Pi and motor driver were combined onto a custom PCB. Below is a schematic showing the circuitry on the custom PCB.



The labels beside the header pins reveal the function of the pins. For example, the pins labeled TO_AMPLIFIER are used to route the 5VDC to the class D amplifier. The pins labeled USB_A were connected to a female USB type A breakout board. A USB to mini-USB cable provided power to the Raspberry Pi. This approach of supplying power to the Pi was selected over using the Vin pins because safety circuitry is embedded in power provided through the female mini-USB port on the Raspberry Pi 0 w.

Software Architecture and Hierarchical Design

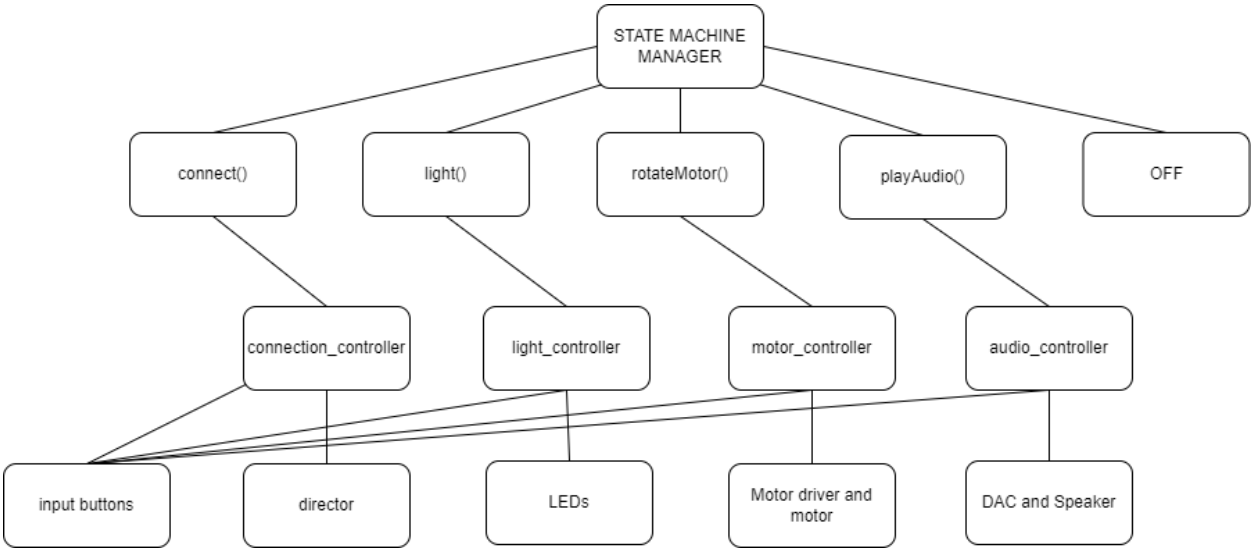
The software is broken up into different submodules: connection_controller, audio_controller, motor_controller, light_controller and main_controller.

Following the issues that were raised about “director connect”, the requirements for the demo excluded the connection_controller, thus testing for this submodule was reduced.

The submodules were first tested individually through manual calls from the main function. After integrating the UI (buttons), we were able to replace the manual calls with a while loop that waited for the user to press a button.

UI-Software Integration

The UI-Software integration can be seen in the block diagram below. Users can press buttons to trigger different robot functions. The logic that links a button to its function can be found in the main controller. Each function has its own software submodule, all of which are described below.



Audio

The audio_controller module has one main function - playAudio() - which takes in the path to a wav file and converts it to an array of integers that is sent via the SPI protocol. This stream is then sent to the DAC, then to the amplifier and finally to the speaker to output audio. The audio_controller is imported in the main controller, and the playAudio function is called when the button for testing audio is pressed and when the whole robot is tested.

Motor

The motor_controller module has one main function - rotateMotor() - which takes in a direction as a string: clockwise, counterclockwise or stop. Given this input, the function will rotate (or stop) the servo motor. The motor_controller is imported in the main controller, and the rotateMotor function is called when the button for testing movement is pressed and when the whole robot is tested.

Lights

The light_controller module has two main functions – lightsOn() and lightsOff(). As the name of the functions indicate it, they turn on or off the LEDs on the robot. The light_controller is imported in the main controller, and the lightsOn/lightsOff functions are called when the button for testing lights is pressed and when the whole robot is tested.

Main

The main controller logic is found in the state machine below. A while loop waits for user input (idle state). If one of the test buttons is pressed, the test is executed, and the code returns to idle. On demo, the code sequentially connects to the director (this was bypassed for the actual demo because of the issues with director connect), plays the audio file (path should have been provided by director, manually input it for demo), turns the LEDs on, and starts rotating the motors. When the audio stops playing, the LEDs and motor are turned off.