Yusuf Shaikh

RedID: 816921177

Compe 160 Section 2 Ken Arnold

December 9, 2013

Student Final Project

**Goal of The Project**

The goal that we projected to find was a way to implement a radio server in our Raspberry Pi and then implement the basic commands of Pause, Play, Next, Previous through GPIO Buttons. The radio stations were available to us from online server links. SO the only problem was getting the software to play the music, wiring of the buttons, coding the direct the buttons to the basic commands and asserting those all into one program.

**Parts:**

* 1: Twin Industries BreadBoard with 460 Total Tie Points
* 4: Mini Push Button Switch
* 6: 6’’ M/F Premium Jumper Wires
* 3: 6’’ M/M Premium Jumper Wires
* 4: 10K-OHM, ¼ -WATT, 5%, Resistors
* 1: Raspberry with Power Outlet
* 1: Monitor with speakers & Power Outlet
* 1: Mouse
* 1:Keyboard
* 1: Ethernet Cable

**Steps:**

1. First we had to assert ourselves to obtain the software/libraries that would play the music retrieve the music and sound for the RPI. We did this through three commands. Just like in class where we obtained the library wiringPi we had to use the get-apt command. We first obtained MPD and MPC. MPD stands for Music Player Demean, a software the integrates like a media player on the pi through the terminal command. The code to obtain this was “sudo apt-get install mpd”. After doing the necessary waiting the software was downloaded and up to date on the Pi. Next we had to obtain the MPC software, which allows us to obtain the servers from online web portals and converts them to a readable binary language that MPD could use the play songs. The command for this was similar to the previous which is “sudo apt-get install mpc.” Finally we obtain Alsa, software that allows us to project volume from the Pi. Thinking of the Pi as a regular computer we would think it would already have something like this on it. However we found that it depends on the OS we originally put on. The simple command for this was sudo “apt-get install alsa-utils”.
2. Next we get to the wiring. The reason that wiring came before the programing part of the lab, was the reason that we had to correlate the GPIO buttons the RPI pins, which will then be used as input monitors on the PI. After looking at the previous wiring for the Button Led. We saw that the resistors had to be in series while LED and buttons in parallel. We then slowly understood the fact that every button and it must share the voltage with the ground. We attached a wire directly to the board representing Voltage then shared it with the 4 buttons simply by adding 10K-OHM resistors from that point to the Button. We did the same concept with the Ground. We had one wire coming directly from the Pi representing ground. Then shared with each button through a string of male to male wires. We then come to the part of GPIO button and pins. I laid down the buttons on the board then watched how the worked diagonally (parallel) I then looked on the GPIO PIN connector chart and correlated the Pins to the proper buttons. After it was all connected I wrote down each GPIO Pin number for each button to use in the program.
3. We then get into the program. Though simple the program had to multiple problems when multiple buttons intact the system. Either one more two would work or neither. But never all four at the same time. We simply implement a basic command of DigitalRead(Pin1) == 1 then project the command MPC play or pause or next. Depending on the function of the button. The final problem occurred when the buttons wouldn’t make anything occur. After a series of debugging trials we realized the code must have had to originally start off. Meaning LOW.
4. Implementing was the easiest and best jobs of the entire setup. We just saved the program as “finalmusic” then implemented it through “gcc – Wall –o music finalmusic.c –lwiringPi” to compile the code altogether. Then we run the program that we coded by typing “sudo ./music” in the terminal. Once that is done, you are ready and able to test each individual button. As a result; the “play” button”, “stop” button, “next” button and the “previous” button all function correctly in their purpose of having an internet radio station audio player.

**The Code**

#include <wiringPi.h>  
#include <stdio.h>  
#include<stdlib.h>  
// First we stated all the Libraries we are going to use throughout the Lab. We mention the wiringPi Library which correlates to the PinMode and other GPIO  commands  
// Then we mention stdio which is the simple library for basic if, while and print statements.  
  
#define pin1 6  
#define pin4 5  
#define pin5 1  
#define pin6 4  
// We defined a couple of variables at this point. We did not necessarily need to do this however it felt to us as it would become easier to us when  we  
// go to complicated parts and there would be just random  numbers written at points than  "pin1". However later realizing the numbers were not correlating with  
// the definitions we just changed the numbers and left the name the same. Saving us time however looking disorganized.  
  
int main (void)  
{  
  int state = 0;  
  wiringPiSetup();  
  pinMode(1,INPUT);  
  pinMode(4,INPUT);  
  pinMode(5,INPUT);  
  pinMode(6,INPUT);  
// Finally we get to the start of the program. We start it of the main function, then define an integer state that will tell us the current state of the program   
// whether it is playing music of is stopped. Then we set 4 pinModes that tells us that those buttons are assigned to a certain GPIO from the RPI which provide us  
// with an input and we can correlate those to a command.  
  
  for(;;)  
  {  
// This for loop is just for the program to continue forever at least until the exit command is called.  
  
        if (digitalRead(pin1) == LOW)  
        {  
  
              delay(500);  
  
                if (state == 0)  
                {  
  
                  system("mpc play");  
                  state = 1;  
  
                }  
  
        }  
//This first "if statement" is there  to show once the music is off and the first button (pin1) is  pushed it will prompt the RPI to start eh command  MPC play.  
// MPC is a software we received from the internet which lets us play internet radio from any station that provides us with a direct server link. and the   
// play option lets us play the music.  
  
        if (digitalRead(pin5) == LOW)  
        {  
  
                delay(500);  
                if(state == 1)  
                {  
  
                  system("mpc stop");  
                  state = 0;  
  
                }  
        }  
// We then get into the second "if" which lets  the user prompt the second button (pin5) the stop the occurring music playing. We did this by setting a state of the  
// RPI which shows weather the RPI is playing music or not at the current moment. If it is then the Stop command will be available if not, then it won't be.  
  
        if (digitalRead(pin4) == LOW)  
        {  
  
                delay (500);  
  
                if(state == 1)  
                {  
  
                system("mpc next");  
  
                }  
  
        }  
// The third if statement  is for the next station. We have incorporated three stations from the internet which then lets us play music. SO by pressing the next   
// button (pin4) it allows us to change to the next station on the playlist  
  
        if (digitalRead(pin6) == LOW)  
        {  
                delay (500);  
  
                if(state == 1)  
                {  
  
                system("mpc prev");  
  
                }  
        }  
// Finally the last if statement is there to go the previous station. We tried to create a MP3 like project. In which we run multiple playlist servers then   
// we could go to the next station and previous station by pressing different buttons. The button you would have to press would be the last one (pin6).  
  
  }  
  return 0;  
}