**COSC 439 - Final Project**

**Team Members**

Michael Santoro Cesar Suchite

Esmir Lugonjic Nick Hess

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**README**

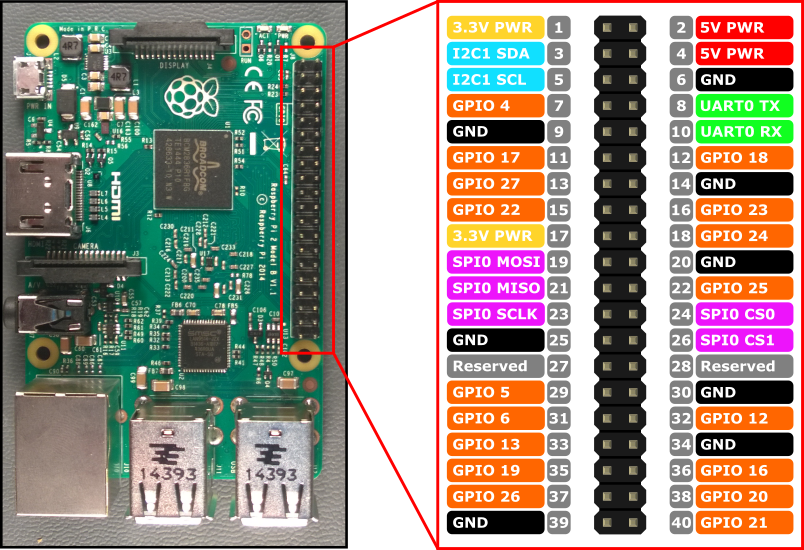
This project utilizes a linux operating system and c programming language to connect a small computing device and a joystick input device in order to play games on an emulator. Our team has successfully implemented this game emulator and we are now able to use our joystick both as input for the general operating system and as input for old time games running through the emulator.

For reference, we are turning in with our readme file, the sample c code create for the joystick driver, and our powerpoint explaining our process. This file will get into the most detail of the complete process of implementation on the raspberry pi.

First of all, we are utilizing the Raspberry Pi 3 Model B for the project. To get the device setup, we first had to download the disc image of the Raspbian operating system, and flash this image to our 16gb SD card. The SD card acts as the hard drive for the computing device. To write the operating system image onto this SD card we inserted the SD card into a windows 8 machine and used the downloaded program “Win32DiskImager”. Now that the operating system had been installed, we placed the card back into the Raspberry Pi and booted the device. For reference, we used a USB keyboard and USB wireless mouse to control the device throughout installation. A final step to this process was expanding the disk space that the operating system partition was utilizing. We had to do this as we ran into a problem of running out of storage on the SD card. This is because the Raspbian image is designed for a 4GB card, and not our 16GB card. This was easily fixable using the command “rasp-config” and the associated menu for expanding the file system.

The next step once we confirmed that the device was working properly, was to begin installing the emulator. We downloaded and unzipped the MAME4All emulator. Download Link can be found here: <https://code.google.com/archive/p/mame4all-pi/> After making a few adjustments, all that was left was to create the necessary folders and run the install that came with the MAME zip folder. Following this, we started up the emulator and confirmed that it was working with our machine. As we show in the demo, to start up the emulator we simply open a linux command prompt and type the path to our mame executable file. So we execute the following command to launch the emulator “/usr/local/bin/indiecity/InstalledApps/mame4all\_pi/Full/mame”. The emulator was working fairly well so now we moved on to find appropriate games for our joystick input.

So to run games on this emulator we needed to download a few ROMs. The free game we utilized in our demo was downloaded from the following link...http://mamedev.org/roms/hardhat/ . This game has joystick input only so it fit our specific needs. We simply downloaded the game and placed the zip folder in the games folder under where our emulator had been installed. We again started up the emulator to make sure it was recognizing our ROM and it was...so now we moved on to implementing the C code that acts as a driver for the GPIO joystick.

So to give some basic information, the Raspberry Pi has an assortment of GPIO pins. These pins have no specified purpose, at least the most of them, and they allowed us to connect our joystick. We referenced the appropriate layout of GPIO pins for our model of the Raspberry PI provided below. <https://ms-iot.github.io/content/en-US/win10/samples/PinMappingsRPi2.htm> 

So when we implemented the C code...this mapping helped us address which pins were supposed to function in which direction, as well as giving us an idea of where the ground pins were and such. The c code that acts as a driver for our joystick, is an adaption from two of our referenced resources provided at the bottom. Our code sets up the mapping of our pins, contains methods to recognize the board and setup, and then other than that the main strategy for our driver is using interrupts for input recognition, as well as implementing a form of timing out so the OS can go back and forth. This beginning set up of the raspberry pi and then implementing and understanding how the source code should bind together with our joystick, took the bulk of the time for this project.

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**PROJECT BREAKDOWN**

Michael Santoro → Responsible for significant implementation of the source code; creation of the demonstration video; creation of the read\_me file; supplied images.

Esmir Lugonjic → Responsible for providing the game to be tested, along with its instructions; partial implementation of the source code; creation of the powerpoint demonstration.

Cesar Suchite → Responsible for partial implementation of the source code;

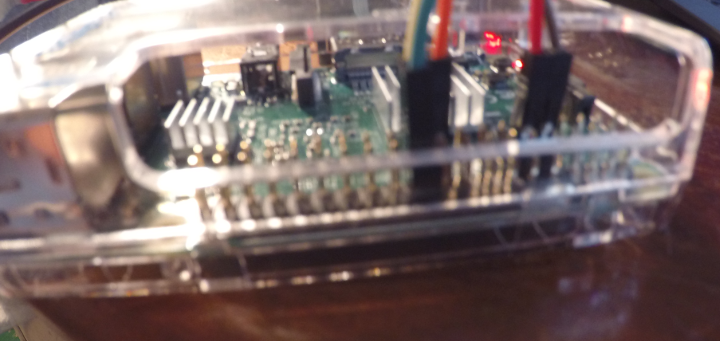
Nick Hess → Responsible for partial implementation of the source code;

Shared Responsibility → All group members contributed to the documentation and presentation materials as well as any necessary hardware and software complications that arose while using and modifying the raspberry pi and associated programs.

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Screenshots**

**Raspberry Pi GPIO pins**



**Raspberry Pi and Joystick**



**Raspbian running on our device**



**Game running through emulators**



////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Link to Demo Video**

[**https://www.youtube.com/watch?v=3ap5blo7yvQ&feature=youtu.be**](https://www.youtube.com/watch?v=3ap5blo7yvQ&feature=youtu.be)

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Lessons Learned**

Overall as a team we realized that the logic even to implement a joystick in the linux operating system is complicated and requires careful thought. We have gained a better understanding of the linux operating system and C programming language by working on this project, and we believe the project is a good final way to wrap up our knowledge of operating systems and the associated implementations and algorithms needed for I/O.

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Referenced Resources**

<https://github.com/recalbox/mk_arcade_joystick_rpi>

<https://learn.adafruit.com/retro-gaming-with-raspberry-pi/buttons>