Force Sensitive Dance Pad

By
Sara Akgul
Nick Birnberg

TA: Henry Duwe

2 March 2016
Project No. 35
Contents

1 Introduction ......................................................... 1

1.1 Statement of Purpose ............................................. 1

1.2 Objectives ......................................................... 1

1.2.1 Goals and Function ........................................... 1

1.2.2 Benefits and Features ....................................... 1

2 Design ........................................................................ 2

2.1 Block Diagrams ..................................................... 2

2.2 Block Descriptions ................................................ 2

2.2.1 Panel Assembly .................................................. 3

2.2.2 Panel MCU ....................................................... 4

2.2.3 Force Sensors ..................................................... 4

2.2.4 LED Display ..................................................... 4

2.2.5 Central Raspberry Pi ............................................ 5

2.2.6 Power Supply .................................................... 6

2.3 Software ............................................................. 8

2.3.1 MCU (Play) ....................................................... 8

2.3.2 MCU (Record) ................................................... 9

2.3.3 Calibration Software .......................................... 10

2.3.4 Feedback ........................................................ 11

2.3.5 Driver ............................................................ 12

3 Analysis .................................................................. 13

3.1 Calculations ......................................................... 13

3.2 Plots ..................................................................... 14

4 Cost ........................................................................ 16

4.0.1 Labor .............................................................. 16

4.0.2 Parts ............................................................... 16

4.0.3 Total ............................................................... 16

4.1 Schedule ............................................................. 17

4.2 Safety ................................................................. 17

4.3 Code Of Ethics ..................................................... 17

Reference ................................................................. 19
1 Introduction

1.1 Statement of Purpose

Dance Dance Revolution and other dance simulation games are an entertaining way to get active and stay in shape. Beginners and professionals develop hand-eye coordination, rhythm, strength, and endurance with regular play, but only when the dance pad feels like its working properly. However, many people still lose interest, and become frustrated. This is because present day DDR pads rely solely on mechanical switches (no other type in production) for input sensing, resulting in poor hardware reliability, inconsistent input sensitivity, and lack of standardization. Our system, based on piezoelectric sensors, would eliminate these problems, integrate easily with current technology, and introduce many new features.

1.2 Objectives

1.2.1 Goals and Function

• Provide consistent, predictable behavior, from pad to pad
• Eliminate the need for constant imprecise non-standardized mods
• Reduce player frustration by using sensors that dont require regular maintenance
• Increase success of competitive events and increase community morale
• Improve the dance game community with a stable standardized platform

1.2.2 Benefits and Features

• Manually adjustable force threshold
• Automatic force threshold calibration by analyzing a players style
• Mode for standardized force threshold value – great for competition
• Easy to install in existing arcade pads and DIY homepad designs
• Works out of box with existing open source Stepmania 5.0 software
• Informative colored LED feedback analogous to force exerted in real time
• Can be used for any dance game with any number of panels due to modularity
• Can be used as generic USB keyboard/joystick input
• Feedback based on foot placement triangulation and force data
• Sensor easy auto-calibration
2 Design

2.1 Block Diagrams

Figure 1: Device block diagram.

2.2 Block Descriptions
2.2.1 Panel Assembly

Figure 2: Panel circuit diagram.
The dance pad consists of four modular panel assemblies. Each panel assembly operates semi-independently. In theory, we can have any number of panel assemblies connected to a central MCU, depending on how many buttons of a dance game a player wants to use.

### 2.2.2 Panel MCU

Directly handles control of the LED Display, decides binary directional triggering in real time based on directions from Central MCU and force profile from force sensor. The panel MCU samples and records the voltages over time of the force sensors, while doing real-time digital signal processing to interpret if the panel was hit within the specified force threshold as well as which color to flash the LEDs. It communicates each hit to the central MCU, while stored voltages are communicated at the end of each song for analytics. It will also be responsible for the initial sampling of the raw force values from the sensor, at about 2kHz, which is sufficient for this application.

#### 2.2.3 Force Sensors

We will be using TekScan A201 sensors[1], which can detect forces from 1-100lb, with the option of changing the range to 1-1000lb if necessary. Typically, the sensor will perform with \( \pm 3\% \) linearity error (with line drawn from 0 to 50% load) and repeatability within 2.5% of Full Scale (Conditioned Sensor, 80% of Full Force Applied). The dynamic range of this force sensor can be modified by bringing the drive voltage input to the sensor anywhere from .1-.15 volts, and adjusting the resistance of the feedback resistor, down to a minimum of 1kΩ. The latency of sensor response is on average 5 microseconds.

#### 2.2.4 LED Display

Multicolored LEDs in pads will light up based on the amount of force exerted on a panel in real-time during play, giving instant feedback to the player. The lights will differ color based upon the amount of force

![Figure 3: TekScan A201 Inverting Op-Amp Circuit.](image-url)
received. The pad will light up green if the players force is within the specified force threshold, red if the player is below the threshold, and violet if the player exceeds the threshold. There will also be an option for absolute force colors, to be able to visually compare forces from panel to panel on a single dance pad assembly.

2.2.5 Central Raspberry Pi

![Raspberry Pi GPIO Pinout](image)

This controller is in charge of issuing control commands to the variable/desired number of panel assemblies, and passing up both raw voltage over time data and directional binary values to software from the panel assemblies up to software for data processing, calibration calculations, and gameplay. USB is used to communicate between the Central MCU and a computer running client software. We need to change the resolution of our data collected by the panel MCUs to be of a manageable size, yet still providing useful information. We will shrink the resolution of the signals from 1024 bits to 32 bits here. Bottleneck for data throughput will be 2Khz sampling with 32 bit resolution, resulting in a approximately $32 \times 2000 \times 12 \times 180 = 16MB$ of raw data on average per song played, which will be stored on the SD card and quickly transferred over USB for the host to process. USB + Arduino latency is low, and throughput is good, with low latency.
2.2.6 Power Supply

Per panel power requirements are in Table 1. Because our design is modular, we want to be able to drop in any number of panel assemblies for a given use case. So, we evaluate the power requirements for the pieces of this module together. For our final demonstration, we will be demoing a dance pad with 4 total panel assemblies, with power requirements shown in Table 2.
Table 1: Power requirements per panel assembly.

<table>
<thead>
<tr>
<th>Part/Module</th>
<th>Voltage Req.</th>
<th>Current/one</th>
<th>Current/all</th>
<th>Power Consumption (P = i * V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEDs (3)</td>
<td>3.3 V</td>
<td>20 mA</td>
<td>60 mA</td>
<td>.198 W</td>
</tr>
<tr>
<td>TekScan A201 Sensors (4)</td>
<td>3.3 V / -1 V</td>
<td>2.5 mA</td>
<td>10 mA</td>
<td>.04 W</td>
</tr>
<tr>
<td>MCP6004 (4)</td>
<td>3.3 V</td>
<td>100uA</td>
<td>400 mA</td>
<td>Negligible</td>
</tr>
<tr>
<td>50kg Load Cell (1)</td>
<td>3.3 V</td>
<td>3.3 mA</td>
<td>3.3 mA</td>
<td>.01 W</td>
</tr>
<tr>
<td>SD Card (1)</td>
<td>3.3 V</td>
<td>50 mA</td>
<td>50 mA</td>
<td>.165 W</td>
</tr>
<tr>
<td>ATMEGA328P (1) [2]</td>
<td>3.3 V</td>
<td>1 A [1]</td>
<td>1:00 AM</td>
<td>3.3 W</td>
</tr>
<tr>
<td><strong>Total Requirements</strong></td>
<td><strong>3.3 V/-1 V</strong></td>
<td><strong>1.13 A</strong></td>
<td><strong>1.13 A</strong></td>
<td><strong>3.713 W</strong></td>
</tr>
</tbody>
</table>

Table 2: Power requirements for entire dance pad.

<table>
<thead>
<tr>
<th>Part/Module</th>
<th>Voltage Req.</th>
<th>Current/one</th>
<th>Current/all</th>
<th>Power Consumption (P = i * V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Panel MCU (4)</td>
<td>3.3V/-1V</td>
<td>1.13A</td>
<td>4.52 A</td>
<td>14.852 W</td>
</tr>
<tr>
<td>Raspberry Pi B+</td>
<td>5V</td>
<td>1A</td>
<td>1:00 AM</td>
<td>5 W</td>
</tr>
<tr>
<td>4:1 Mux/Demux</td>
<td>3.3V</td>
<td>3.3 mA</td>
<td>3.3m A</td>
<td>.01 W</td>
</tr>
<tr>
<td><strong>Total Requirement</strong></td>
<td><strong>5V/3.3V/-1V</strong></td>
<td><strong>6.53 A</strong></td>
<td><strong>6.53 A</strong></td>
<td><strong>19.862 W</strong></td>
</tr>
</tbody>
</table>
When a song has begun, the PC software sends a signal to the central Raspberry Pi. The Raspberry Pi then sends a command message (0x1 - Play) to each panel MCU. The panel MCU’s begin recording, and sending interrupts on registered hits, which get quickly interpreted as arrow hits through the driver. The MCU is
stuck in this loop until the song is over (the serial connection between the central Raspberry Pi is set to high). The MCU is then reset to Command Mode, and awaits a Collect Data message (0x4), transmits all the data collected, and then clears the SD card.

2.3.2 MCU (Record)

![Diagram of MCU record command]

Figure 7: MCU record command.

The record command is given to the MCU during the calibration process in Section 2.3.3. It individually samples each sensor and then immediately sends the voltage information back to the central Raspberry Pi.
2.3.3 Calibration Software

Figure 8: PC Software - Calibration

- Prompt user to input weight
- Prompt user to step on one panel.
- Repeat for all panels.
- Record analog values for each pad.
- Prompt user to step on left and right panels.
- Repeat for top and bottom panels.
- Record analog values for each pad.
- Prompt user to step off the pad.
- Communicate scale values to each MCU.
- Record analog values for each pad.
- Calculate scale values for the sensors.
Initial Calibration sequence for the sensors upon startup of the pad for consistent sensitive for a given set of force threshold values from one physical pad or panel to another. This will ensure that pads loaded with the same settings will always feel the same no matter what.

For our automatic adjust-to-player mode, the software will analyze raw force data during play and automatically calibrate the force thresholds by learning the players technique and tendencies. This will be done by entering a special software mode, which enables the dance pad to start varying force thresholds an experimentally determine the optimal force values with repeated play and analysis.

Also will allow custom force threshold values to be entered, copied from a past stored configuration, and also validate a players eligibility for competition on songs completed by checking standardized force thresholds are loaded into the pad.

2.3.4 Feedback

![Feedback Diagram]

Figure 9: PC Software - Feedback

The portion of software responsible for feedback will help the user figure out what they can do to improve their playing. We have two main types of feedback that software must handle.

**Realtime feedback:** LED Feedback - Light LEDs for each panel for specified force thresholds.
Analytics:


2. Distance Charts - Identify pre-programmed maneuvers during the songs where a graph depicting the distance from the center would be useful, and generate them based on sensor data.

2.3.5 Driver

Handles the low level USB communication between the dance pad and PC, communicates sensor threshold mode of operation (loaded force profile, initial calibration mode, adjust-to-player mode), communicates LED data and LED mode of operation (amount of force mode, target force feedback), establishes and manages connection between Central MCU and PC.
3 Analysis

3.1 Calculations

Load Cell Initial Calibration Procedure:

The following calculation is very important in the initial calibration of our sensors in order to properly map analog to digital converted sensor readings to real life force values to ensure consistency of signal response in our dance pads. These sensors have dynamic ranges, and vary from one to another. To do this, we employ the method given by the manufacturer [1]. For the following test weights and force readings, we can derive a linear relation to assign arbitrary force inputs on the sensor, weights 0-100lb, sampled with 10-bit resolution, as follows, for sample data points:

Line equation derivation: slope = rise/run = 30-10/(347-102) = 12.25

\[ y = 12.25x + b, \quad 347 = 12.25(30) + b, \quad b = -20.5. \]

Equation for conversion is now \[ y = 12.25x - 20.5. \]

So, for a force of 35lb exerted on a standalone sensor, we should expect to get an A/D converted signal reading of \[ y = 12.25(35) - 20.5 = 408.25 \] which is about 408. Now, we can test this weight and verify the repeatability and accuracy of our conversions, as well as approximate force values within a guaranteed margin of error by the sensor manufacturer.

A/D Resolution + Data Size Analysis

In the above sample calculation for sensor calibration using two test weight points, we use A/D values with 10-bit resolution. However, for the actual implementation in our circuit, we need to strike the balance between resolution of the data and not having too much data to copy over at the end of a song play loop. So, we chose 6-bit resolution, which scales our A/D force values of 1-100lb per sensor to values of 0 to 63. In the worst case scenario, we can expect a song that is played to be around 6 minutes long. Because the tightest timing window of accuracy for the game is 21.5ms, with 10.5ms early/late, a sampling the sensor signals every 2 ms during game play will be sufficient for our data collection purposes.

So, we have 6 bits of data, every 2 ms, at a worst case scenario of 6 minutes of data collection, per sensor, of which we have 16. So, the total size of data we will need to send at the end of a song, in the worst case scenario, will be:

\[ 6 \text{minutes} \times 60 \text{seconds/minute} \times 1/0.002 \text{seconds} \times 16 \times 6 = 2.16 \text{ Megabytes of data per song.} \]

Because we are sending this data over serial and USB, the time to transfer 2.16 Megabytes of data and do post-processing will be negligible.

Real Time Constraint Concerns of the System

For real time force triggering constraints, we have chosen to have two separate methods of handling tap inputs. An interrupt driven approach, where each panel MCU pre-empts the central MCU (raspberry pi) on a tap hit, which would easily occur within the timing constraint (would take nanoseconds to execute because of 8 MHz + clock speeds.) Separating this from the mass data collection and analysis guarantees acceptable real time input response from the system.
3.2 Plots

Testing was done in order to determine the profile of the signal for a tap step on uncalibrated Tekscan A201 Flexiforce sensors. This analysis is important for verification of the resolution and sampling rate of the A/D conversion of sensor output. Figures 10 and 11 are plots created based on .csv data generated from the following arduino sketch:

```c
extern "C" {
    #include "Arduino.h"
}

const int fsrPin = A0

void setup() {
    pinMode(fsrPin, INPUT);
    Serial.begin(250000);
}

void loop() {

    Serial.print(analogRead(fsrPin));
    Serial.print(" ", "");
    Serial.print(millis());
    Serial.println();
    delay(2);
}
```

Figure 10: Simulated tap sensor readings.
Figure 11: Cleaned sensor readings
4 Cost

4.0.1 Labor

Table 3: Labor Costs

<table>
<thead>
<tr>
<th>Name</th>
<th>Hourly Rate (USD/hour)</th>
<th>Hours</th>
<th>Total = ($/hour) x 2.5 x hours to complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nick</td>
<td>40</td>
<td>250</td>
<td>$25,000</td>
</tr>
<tr>
<td>Sara</td>
<td>40</td>
<td>250</td>
<td>$25,000</td>
</tr>
</tbody>
</table>

$50,000

4.0.2 Parts

All parts are to be ordered off DigiKey and SparkFun.

Table 4: Parts Costs

<table>
<thead>
<tr>
<th>Part</th>
<th>Manufacturer</th>
<th>Retail Cost ($)</th>
<th>Quantity</th>
<th>Actual Cost ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A201</td>
<td>TekScan</td>
<td>14.63</td>
<td>16</td>
<td>234.08</td>
</tr>
<tr>
<td>ATMEGA328P</td>
<td>Atmel</td>
<td>3.70</td>
<td>4</td>
<td>29.60</td>
</tr>
<tr>
<td>LM7805CT</td>
<td>Fairchild Semiconductor</td>
<td>0.62</td>
<td>4</td>
<td>2.48</td>
</tr>
<tr>
<td>Raspberry Pi 2 Model B</td>
<td>Raspberry Pi Foundation</td>
<td>35.00</td>
<td>1</td>
<td>35.00</td>
</tr>
<tr>
<td>16 MHz Crystal</td>
<td>Citizen Finedevice Co Ltd</td>
<td>0.54</td>
<td>4</td>
<td>2.16</td>
</tr>
<tr>
<td>Red LED</td>
<td>Cree</td>
<td>0.32</td>
<td>1</td>
<td>0.32</td>
</tr>
<tr>
<td>Blue LED</td>
<td>Cree</td>
<td>0.32</td>
<td>1</td>
<td>0.32</td>
</tr>
<tr>
<td>Violet LED</td>
<td>Cree</td>
<td>0.32</td>
<td>1</td>
<td>0.32</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>310.38</td>
</tr>
</tbody>
</table>

4.0.3 Total

Table 5: Total Costs

<table>
<thead>
<tr>
<th>Section</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labor</td>
<td>$50,000</td>
</tr>
<tr>
<td>Parts</td>
<td>$310.38</td>
</tr>
<tr>
<td></td>
<td>$50,314.84</td>
</tr>
</tbody>
</table>
4.1 Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Task</th>
<th>Assignee</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/7</td>
<td>Complete and test Raspberry Pi Plug &amp; Play Driver</td>
<td>NB</td>
</tr>
<tr>
<td></td>
<td>Complete and test breadboard assembly of sensors</td>
<td>SA</td>
</tr>
<tr>
<td>3/14</td>
<td>Finish and Test Single Panel Assembly</td>
<td>SA</td>
</tr>
<tr>
<td></td>
<td>Finish and Test Basic Analytics Software</td>
<td>NB</td>
</tr>
<tr>
<td>3/21</td>
<td>Company Retreat</td>
<td>NB, SA</td>
</tr>
<tr>
<td>3/28</td>
<td>Finish and test Calibration software</td>
<td>SA</td>
</tr>
<tr>
<td></td>
<td>Complete all MCU software</td>
<td>NB</td>
</tr>
<tr>
<td>4/4</td>
<td>Finish Calibration Software</td>
<td>SA</td>
</tr>
<tr>
<td></td>
<td>Test all MCU software</td>
<td>NB</td>
</tr>
<tr>
<td>4/11</td>
<td>Focus on User Experience on Client Software</td>
<td>NB</td>
</tr>
<tr>
<td></td>
<td>Integrate Circuitry With Dance Pad Hardware</td>
<td>SA</td>
</tr>
<tr>
<td>4/18</td>
<td>Testing and Debug Client Software</td>
<td>NB</td>
</tr>
<tr>
<td></td>
<td>Test and Debug circuitry</td>
<td>SA</td>
</tr>
<tr>
<td>4/25</td>
<td>Complete software sections of final paper</td>
<td>NB, SA</td>
</tr>
<tr>
<td></td>
<td>Complete circuitry sections of final paper</td>
<td>SA</td>
</tr>
<tr>
<td>5/2</td>
<td>Focus on presentation aesthetics</td>
<td>SA</td>
</tr>
<tr>
<td></td>
<td>Focus on final paper proofreading and aesthetics</td>
<td>NB</td>
</tr>
</tbody>
</table>

4.2 Safety

The Force Sensitive Dance Pad consists mainly of digital logic operating at a low voltage, completely enclosed in a dance-pad fixture. The dance-pad fixture itself provides the regulated voltage lines. Thus there are 2 primary concerns:

1. Users must handle the dance-pad fixture to its operating instructions.
2. Users must not wear footwear that have a hazardly low coefficient of friction, as they risk slipping while using the device.
3. Users must be healthy enough for physical activity.

Both members of the team have completed the necessary lab safety training before engaging with any lab work.

4.3 Code Of Ethics

We, the members of the IEEE, in recognition of the importance of our technologies in affecting the quality of life throughout the world, and in accepting a personal obligation to our profession, its members and the communities we serve, do hereby commit ourselves to the highest ethical and professional conduct and agree[3]:

17
1. to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;

2. to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist;

3. to be honest and realistic in stating claims or estimates based on available data;

4. to reject bribery in all its forms;

5. to improve the understanding of technology; its appropriate application, and potential consequences;

6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;

7. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;

8. to treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression;

9. to avoid injuring others, their property, reputation, or employment by false or malicious action;

10. to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.
References


Appendix A  Requirement and Verification Table

Table 6: System Requirements and Verifications

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microcontroller Unit</strong></td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>1. Digitally processes sensor feedback to detect a hit within 10.5 milliseconds. (a) Has a 2V sensitivity default, thus registers hits above 2V. (b) Does not register a hit when senses 1V.</td>
<td>1. Generate a simulated sensor hit and ensure it is recognized. (a) Set and clear reset (pin 1) to enter command mode. (b) Send play command to command pin (pin 2) at 9600 baud, consisting of bytes Play (0x1). (c) Attach a function generator to analog input pin 23, and set all other analog inputs (pins 24-28) to ground. Attach to channel 1 on oscilloscope. (d) Attach digital output pin 2 to oscilloscope channel 2. (e) Generate an single-shot 3V, 1 MHz square wave with a 100% duty cycle at provided sensitivity level. (f) Ensure that the output of channel 2 is high within 10.5 milliseconds of channel 1 being high. (g) Repeat with a 1V square wave. Ensure output of channel 2 remains low.</td>
<td></td>
</tr>
<tr>
<td>2. When sent a message specifying minimum threshold as 2.5V, correctly processes hits for voltages above threshold and disregards hits for voltages below threshold.</td>
<td>2. Set sensitivity levels and ensure hits are detected accordingly. (a) Set and clear reset (pin 1) to enter command mode. (b) Send sensitivity command to command pin (pin 2) at 9600 baud, consisting of bytes SensitivityMin (0x2), and minimum voltage (0x199). (c) Repeat verification 1 with a 3.5V square wave as an ensured hit, and a 2.5V square wave as an ensured miss.</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
3. When sent a message specifying desired sensitivity range as 2.5V-3V, sets signal for red LEDs if below range, green LEDs if within range, and violet LEDs is over range.

3. Set desired levels and ensure the proper LEDs are selected.
   (a) Attach LED selectors (pins 3 and 4) as separate channels to an oscilloscope. Ensure both pins are low.
   (b) Set and clear reset (pin 1) to enter command mode.
   (c) Send range command to command pin (pin 2) at 9600 baud, consisting of bytes Range (0x3), lower voltage (0x199) and higher voltage (0x267).
   (d) Send play command to command pin (pin 2) at 9600 baud, consisting of bytes Play (0x1).
   (e) Attach a function generator to analog input pin 23, and set all other analog inputs (pins 24-28) to ground.
   (f) Generate an single-shot 2.1V, 1 Hz square wave with a 100% duty cycle at provided sensitivity level.
   (g) Ensure that red LEDs are selected (pin 3 is high, pin 4 is low).
   (h) Repeat for red LEDs are selected (pin 3 is high, pin 4 is low)

LED Display
1. Proper LEDs light up when selected.
   (a) When select signal is 00, no LEDs light up.
   (b) When select signal is 01, red LEDs light up.
   (c) When select signal is 10, green LEDs light up.
   (d) When select signal is 11, violet LEDs light up.

1. Drive demux and observe.
   (a) Attach wires to the demux select pins.
   (b) Attach both wires to ground.
   (c) Ensure no LEDs light up.
   (d) Repeat until all requirements are met by driving wires to a 5V source to simulate a high input.

5

Continued on next page
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
<th>Points</th>
</tr>
</thead>
</table>
| **Analysis Software**  
1. Generate visual heatmap of the players foot placement. | 1. Verification for Item 1:  
(a) Run the client software and click Record Moves  
(b) Place a 2 lb weight in the lower left corner of each panel, then remove and place a 4 lb weight in the lower right corner of each panel, then remove and place a 6 lb weight in the top of each panel.  
(c) Click Stop Recording and Generate Feedback in the client software.  
(d) Ensure that a diagram is generated with a light red color in lower left corner of each panel, a darker shade in the lower right corner of each panel, and the darkest shade of red in the top of each panel. | 10     |
| **Central Unit / Driver**  
1. Plug & Play compatibility.  
(a) Recognized as a USB Joystick when connected to a PC.  
(b) Functions as USB Joystick is expected to. | 1. Connect the Raspberry Pi to a PC.  
(a) Attach a function generator to one of the four digital input pins.  
(b) Generate an single-shot 5V, 1 Hz square wave with a 100% duty cycle.  
(c) Assure that the corresponding arrow key is recorded as pressed on the PC for one second. | 10     |
| **Power Supply**  
1. $V_{out} = 3.3V \pm 0.1V$ at 1 A | 1. Verification process for Item 1:  
(a) Attach 3.3 $\Omega$ Resistor as load  
(b) Attach oscilloscope across load  
(c) Supply regulator with 12 VDC  
(d) Ensure output voltage remains between 3.2 and 3.4 volts | 5      |