Homework 6: Test Plan

T04:

Sean Hendrickson

Waleed Alhaddad

Adrian Steele

Taylor Griffin

November 25, 2015

Version 1.0

Hierarchical Test Plan:

Unit/Module Test

Materials needed:

Main program code: <https://github.com/hsean/ECE-411-Practicum/blob/master/Code/mainProgram.c>

C environment i.e. GCC and Linux

* Main program
  + Angle to PWM function
  + Servo position function
  + Accelerometer data conversion function
  + LED status function

Integration Test

Materials needed:

MMA8452Q Accelerometer

Atmel ATmega328-P

Mini Maestro Servo Controller

HS-422 Servos

DotStar LED strip

Atmel Studio

AVR Dragon Programmer

74ACT125N Buffer

Communications test code: <https://github.com/hsean/ECE-411-Practicum/blob/master/Code/test_communication.c>

* Microcontroller to Servo controller (USART) communication
* Accelerometer to Microcontroller (TWI) communication
* Microcontroller to LED strip (SPI) communication
* Microcontroller level shifting (buffer) timing

Parametric Test

Materials needed:

D24V5F3 voltage regulator

LM2940CT voltage regulator

DotStar LED strip

74ACT125N Buffer

HS-422 Servos

Mini Maestro Servo Controller

HC49US 16 MHz Oscillator

4 AA batteries

Oscilloscope

Multi-meter

Atmel ATmega328-P

* Output voltage from voltage regulators
* Level shifting output voltage
* External oscillator frequency
* LED strip input current
* Battery output current and voltage
* Servo input current
* Range of servos

Functional Test

Materials needed:

MMA8452Q Accelerometer

HS-422 Servos

Mini Maestro Servo Controller

Status LEDs

Atmel ATmega328-P

1 Self Leveling Device

Arduino

Arduino IDE

Arduino accelerometer example code: <https://github.com/hsean/ECE-411-Practicum/blob/master/Code/Example%20Code/Arduino%20Accelerometer/MMA8452Q_BasicExample.ino>

* Verify accelerometer outputs valid measurements
* Verify Servo controller operates servos
* Verify LED strip illuminates
* Verify microcontroller GPIO output for Status LEDs
* Verify servos are operable
* Verify microcontroller operation
* Run full system test leveling test
* Run full system power duration test
* Verify servo adjustment time

Exhaustive Test

Materials needed:

1 Self Leveling Device

Atmel Studio

AVR Dragon Programmer

Exhaustive test code: In progress

* Complete test of all possible servo positions with tray
* Complete test of all possible angle to PWM values

Stress Test

Materials needed:

1 Self Leveling Device

* Test maximum tray weight

Use Test

Materials needed:

1 Self Leveling Device

* Handheld operation

Error Test

Materials needed:

1 Self Leveling Device

Protractor

* Orientation testing (outside of maximum angle operation ex. Upside down)
* Tray overweight test
* Communication errors

Test Case Descriptions:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Writer: T04 | | | | | | | |
| Test Case Name: | Accelerometer Function Test | | | | Test ID #: | | ACCL-FT-01 |
| Description: | Verify accelerometer outputs valid measurements  Connect Accelerometer to Arduino and verify functionality using example code. Example code is under Code/Example%20Code/Arduino%20Accelerometer/MMA8452Q\_BasicExample.ino | | | | Type: | | Black Box |
| Tester Information | | | | | | | |
| Name of Tester: | |  | | | Date: | |  |
| Hardware Ver: | | 1.0 | | | Time: | |  |
| Setup: | | Materials:  Hardware: 1 Arduino Uno, 1 MMA8452Q accelerometer on breakout board.  Software: Arduino IDE, code from <https://github.com/hsean/ECE-411-Practicum/blob/master/Code/Example%20Code/Arduino%20Accelerometer/MMA8452Q_BasicExample.ino>  - Connect the GND pin of the accelerometer to the GND pin of the arduino.  - Connect the SCL pin of the accelerometer to a 330 Ω resister and then through to the SCL pin of the arduino.  - Connect the SDA pin of the accelerometer to a 330 Ω resister and then through to the SDA pin of the arduino.  - Connect the 3.3V pin of the accelerometer to the 3.3V output on the arduino.  - Connect Arduino to PC | | | | | |
| Step | Action | Expected Result | Pass | Fail | N/A | Comments | |
| 1 | Compile and Execute Arduino code | IDE should generate no warnings or errors |  |  |  |  | |
| 2 | View serial data stream from Arduino | Outputting 3 values: x, y and z (in degrees) |  |  |  |  | |
| 3 | Place Accelerometer flat | x should be 0° ± 5°, y should be 0° ± 5°, z should be 90° ± 5°. |  |  |  |  | |
| 4 | Align Accelerometer along x axis to 45° | x should be 45° ± 5°, y should be 0° ± 5°, z should be 45° ± 5°. |  |  |  |  | |
| 5 | Allign accelerometer along y axis to 45° | x should be 0° ± 5°, y should be 45° ± 5°, z should be 45° ± 5°. |  |  |  |  | |
| Overall test result: | | |  |  |  |  | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Writer: T04 | | | | | | | |
| Test Case Name: | D24V5F3 voltage regulator Parametric Test | | | | Test ID #: | | VR1-PT-01 |
| Description: | Output voltage from voltage regulator  Testing the output voltage of the voltage regulator. 6 V is applied as input, the datasheet specifies the output voltage should be 3.3 V. | | | | Type: | | Black Box |
| Tester Information | | | | | | | |
| Name of Tester: | |  | | | Date: | |  |
| Hardware Ver: | | 1.0 | | | Time: | |  |
| Setup: | | Materials:  Hardware: 1 D24V5F3 voltage regulator, 1 multi-meter, 1 power supply  - Connect the Vin pin of the voltage regulator to the positive output of the power supply  - Connect the GND pin of the voltage regulator to the negative output of the power supply  - Connect Vout to the multi-meter positive terminal  - Connect the multi-meter negative terminal to ground | | | | | |
| Step | Action | Expected Result | Pass | Fail | N/A | Comments | |
| 1 | Set the power supply to output 5.5 V | Multi-meter reads 3.3 V ± 10 % |  |  |  |  | |
| 2 | Set the power supply to output 6 V | Multi-meter reads 3.3 V ± 10 % |  |  |  |  | |
| 3 | Set the power supply to output 6.5 V | Multi-meter reads 3.3 V ± 10 % |  |  |  |  | |
| 4 | Set the power supply to output 7 V | Multi-meter reads 3.3 V ± 10 % |  |  |  |  | |
| Overall test result: | | |  |  |  |  | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Writer: T04 | | | | | | | |
| Test Case Name: | Oscillator Frequency Parametric Test | | | | Test ID #: | | OF-PT-02 |
| Description: | External oscillator frequency  Test that the external oscillator supplying the clock to the microcontroller has a 16 MHz frequency. | | | | Type: | | Black Box |
| Tester Information | | | | | | | |
| Name of Tester: | |  | | | Date: | |  |
| Hardware Ver: | | 1.0 | | | Time: | |  |
| Setup: | | Materials:  Hardware: 1 HC49US 16 MHz Oscillator, 1 Atmel ATmega328-P, 1 oscilloscope, AVR Dragon Programmer, 1  power supply  Software: Atmel Studio  - Connect pin 9 and 10 (XTAL1 and XTAL2) of the microcontroller to the two pins of the Oscillator  - Connect pin 9 (XTAL1) of the microcontroller to a 22nF capacitor (will be in parallel to the connection to the oscillator)  - Connect pin 10 (XTAL2) of the microcontroller to a 22nF capacitor (will be in parallel to the connection to the oscillator)  - Connect the other sides of the capacitors to ground.  - Connect the microcontroller to the power supply and AVR Dragon Programmer (use the following URL as a wiring guide: <http://www.atmel.com/webdoc/avrdragon/avrdragon.SCKT3200A2.html>)  - In Atmel Studio, set the CKSEL fuses to “0000” to run the microcontroller on the external clock  - Connect the positive end of the oscilloscope probe to pin 9 (XTAL1) of the microcontroller  - Connect the negative end of the oscilloscope probe to ground | | | | | |
| Step | Action | Expected Result | Pass | Fail | N/A | Comments | |
| 1 | Set the power supply to output 3.3 V | On the oscilloscope you should see a clock pulse at 16 MHz. |  |  |  |  | |
| Overall test result: | | |  |  |  |  | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Writer: T04 | | | | | | | | | |
| Test Case Name: | | Angle to PWM Module Test | | | | | Test ID #: | | A2P-MT-01 |
| Description: | | Angle to PWM function  Test that the external oscillator supplying the clock to the microcontroller has a 16 MHz frequency. | | | | | Type: | | White Box |
| Tester Information | | | | | | | | | |
| Name of Tester: | | |  | | | | Date: | |  |
| Hardware Ver: | | | 1.0 | | | | Time: | |  |
| Setup: | | | Materials:  Software: C Environment (ex. GCC and Linux), main program code from: <https://github.com/hsean/ECE-411-Practicum/blob/master/Code/mainProgram.c>  - Extract the “uint16\_t AngleToPWM(int angle)” function from the main program code as well as the #defines that the function uses (MAX\_ANGLE, MIN\_ANGLE, PWM\_MIN, PWM\_MAX) | | | | | | |
| Step | Action | | | Expected Result | Pass | Fail | N/A | Comments | |
| 1 | Supply function with various inputs that are less than the minimum angle allowed. Parameter MIN\_ANGLE should be less than parameter MAX\_ANGLE | | | The function should return -1 |  |  |  |  | |
| 2 | Change MIN\_ANGLE and MAX\_ANGLE parameters to be equal. Supply function with various inputs inside and outside of the acceptable angle range (input < MIN\_ANGLE, MIN\_ANGLE <= input <= MAX\_ANGLE, MAX\_ANGLE < input) | | | The function should return -1 |  |  |  |  | |
| 3 | Change MIN\_ANGLE parameter to be greater than MAX\_ANGLE parameter. Supply function with various inputs inside and outside of the acceptable angle range (input < MIN\_ANGLE, MIN\_ANGLE <= input <= MAX\_ANGLE, MAX\_ANGLE < input) | | | The function should return -1 |  |  |  |  | |
| 4 | Supply function with various inputs that are greater than the maximum angle allowed. Parameter MIN\_ANGLE should be less than parameter MAX\_ANGLE | | | The function should return -1 |  |  |  |  | |
| 5 | Supply the function with inputs beginning at the minimum angle allowed, incrementing by 1 until the maximum angle is reached. Verify that the input matches the desired output. | | | For any given input value, the following ratio should be true:C:\Users\CaptaNran\Documents\Adrian the silent ninja!\College\ECE 411\AngleToPWMverification.png |  |  |  |  | |
| Overall test result: | | | | |  |  |  |  | |