Glovesy

BY

Alan Devine - 17412402

Sean Moloney - 17477122

A Testing Document

As a requirement for CA400

Last Revision: 07/05/2021

Dublin City University (DCU)

PLAGIARISM DECLARATION

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I/We engage in plagiarism, collusion, or copying. I/We have read and understood the Assignment Regulations. I/We have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

I/We have read and understood the referencing guidelines found at
`https://www.dcu.ie/info/regulations/plagiarism.shtml`,
`https://www4.dcu.ie/students/az/plagiarism`,
and/or recommended in the assignment guidelines.

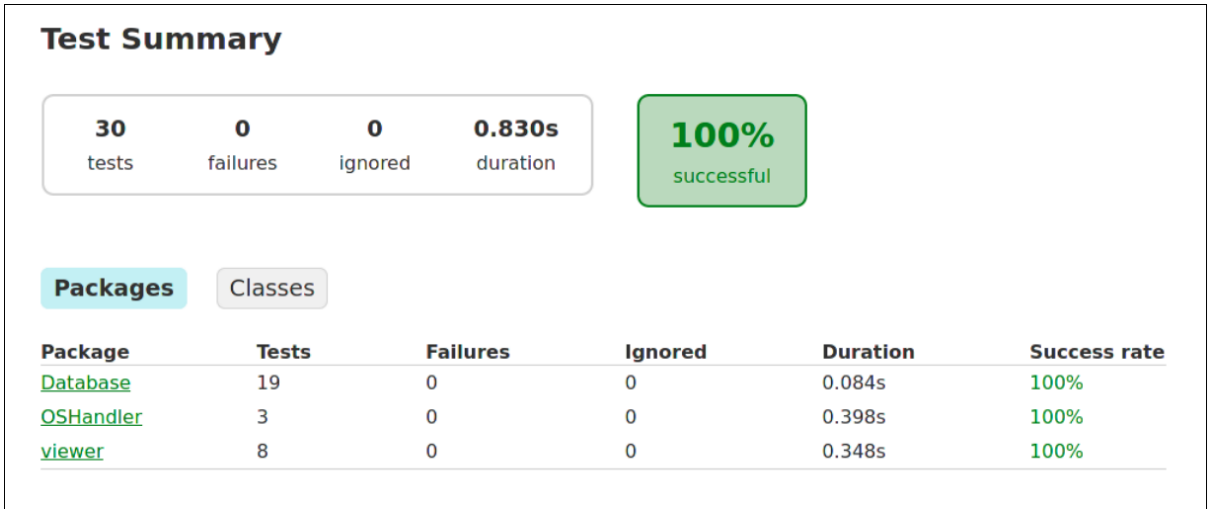
Project Title	Glovesy
By	Alan Devine - 17412402
	Sean Moloney - 17477122
Field of Study	Computer Science
Project Advisor	David Sinclair
Academic Years	2020/2021

TABLE OF CONTENTS

LIST OF FIGURES

Unit Testing

Unit testing was achieved through JUnit5 and executed by Gradle upon each build of the project. Our strategy for writing unit tests went as follows. In addition to running the test suite, Gradle will generate a test result website. The latest test results can be seen in Figure 1.



Test Summary

30	0	0	0.830s
tests	failures	ignored	duration

100% successful

Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
Database	19	0	0	0.084s	100%
OSHandler	3	0	0	0.398s	100%
viewer	8	0	0	0.348s	100%

Figure 1: Unit Test results

Embedded Unit Testing

Unit testing of the embedded systems was carried out by check that the types returned by each function was correct, and that the conversion from voltage to resistance returned the correct value.

```
#include <Adafruit_LSM6DS33.h>

// For SPI mode, we need a CS pin
#define LSM_CS 10
// For software-SPI mode we need SCK/MOSI/MISO pins
#define LSM_SCK 13
#define LSM_MISO 12
#define LSM_MOSI 11

Adafruit_LSM6DS33 lsm6ds33;

void setup(void) {
  Serial.begin(115200);
  testAccelerometer();
  testFlexA0();
  testResistanceConversion();
}

int types(String a) { return 0; }
int types(int a) { return 1; }
```

```
int types(char *a) { return 2; }
int types(float a) { return 3; }
int types(bool a) { return 4; }
int types(uint32_t a) { return 5;}

void testAccelerometer(){
  sensors_event_t accel;
  sensors_event_t gyro;
  sensors_event_t temp;
  lsm6ds33.getEvent(&accel, &gyro, &temp);

  if (types(accel.acceleration.x) == 3) {
    if (types(accel.acceleration.y) == 3) {
      if (types(accel.acceleration.z) == 3) {
        Serial.println("testAccelerometer :\t\tPASS");
        return;
      }
    }
  }
  Serial.println("testAccelerometer :\t\tFAIL");
}

void testFlexA0() {
  if (types(analogRead(A0)) == 5) {
    Serial.println("testFlexA0 :\t\t\tPASS");
    return;
  }
  Serial.println("testFlexA0 :\t\t\tFAIL");
}

void testResistanceConversion(){
  float result = get_resistance(5);
```

```
  if (result == 956920.0) {
    Serial.println("testResistanceConversion: \tPASS");
    return;
  }
  Serial.println("testResistanceConversion: \tFAIL");
}

float get_resistance(int ADCflex) {
  // Convert value from flex sensor to resistance
  float VCC = 3.3;
  int R_DIV = 4700;
  float Vflex = ADCflex * VCC / 1023.0;
  float Rflex = R_DIV * (VCC / Vflex - 1.0);

  return Rflex;
}

void loop() {
}
```

Listing 1: Sample Embedded system unit test

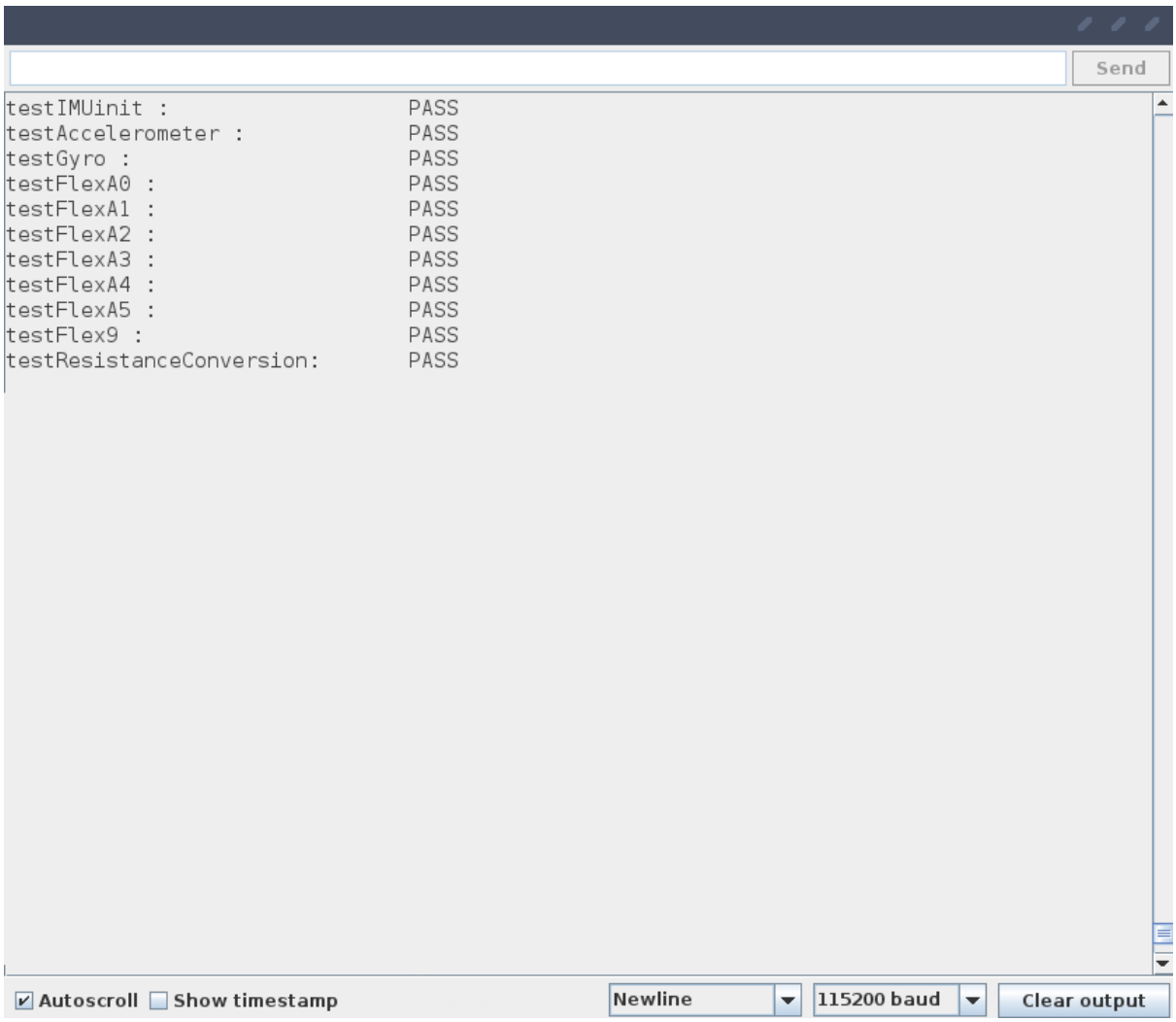


Figure 2: Embedded System Unit test result

Database Testing

Unit tests performed on objects which interacted with our database in some variety made use of a separate Test database which was populated, and cleared in each unit test was run. This is the basic form of a unit test in this package. A Handler object is instantiated in the test database. Then tests are performed on each read and write operation

```
// ApplicationHandlerTest.java
ApplicationHandler db = new
    ApplicationHandler("mongodb://127.0.0.1:27017", "test");
@Test
void addEntry() {
    db.deleteAll();
    db.addEntry(new Document("name", "gcc").append("path",
        "/usr/bin/gcc"));
    assertTrue(db.containsEntry(new Document("name", "gcc")));
    db.deleteAll();
}
```

Listing 2: Application handler test

NOTE: gcc was chosen for a majority of the tests in this suite due to its presence on a majority of Linux distributions.

OSHandler Testing

```
@Test
void parseKeySequence() throws AWTException {
    InputManager inputManager = new InputManager();

    String[] simplePassingKeySequence = new String[] {
            "press,14",
            "release,14"
    };

    String[] simpleFailingKeySequence = new String[] {
            "mash,14"
    };

    String[] nonMatchingKeySequence = new String[] {
            "press,14",
            "press,15",
            "release,15"
    };

    Assertions.assertTrue(
           inputManager.parseKeySequence(simplePassingKeySequence));
    Assertions.assertFalse(
           inputManager.parseKeySequence(simpleFailingKeySequence));
    Assertions.assertFalse(
           inputManager.parseKeySequence(nonMatchingKeySequence));
```

```
}
```

Listing 3: OSHandler Test

Viewer Testing

0.1 Ad-hoc Testing

Due to the nature of the InputManager Class, testing its functionality through unit testing would be, for the most part, unachievable. As such, ad-hoc testing was used to some degree.

0.1.1 executeKeySequence()

I started by making a main method that instantiates the InputManager object and calls executeKeySequence().

```
public static void main(String[] args) throws AWTException {
   // Alt Tab
   String[] sequence = new String[] {
     "press,18"
     "press,9"
     "release,9"
     "release,18"
  };


   InputManager inputManager = new InputManager();
   inputManager.executeKeySequence(sequence);
}
```

Listing 4: ExecuteKeySequence Test

The expected result for this portion of code was for the program to execute alt-tab and for my last focused window to come into view.

0.1.2 moveMouse()

I started by making a main method which instantiates the InputManager object and calls moveMouse().

```
public static void main(String[] args) throws AWTException {
   InputManager inputManager = new InputManager();
   //                      x   y
   inputManager.moveMouse(100, 0);
}
```

Listing 5: moveMouse Function

The result of running this method is the mouse cursor position moving right. In order to verify the distance travelled by the cursor, I used xev to display the cursor position before and after running the program.

In order to test most methods in the class, I made use of the program xev. This program prints the contents of X events, with X being the standard Linux display server. For mouse input a window such as the one below will appear with the output being displayed in the terminal.

Placing the mouse cursor on the window show in Figure 3 and running the program resulted in the output shown in Figure 4. As we can see, there was a change in horizontal cursor position which matches the that which was specified in the main method above.

NOTE: While this is very much a ”synthetic” mouse event, xev will only mark an event as synthetic if such an event is generated by another X11 client.

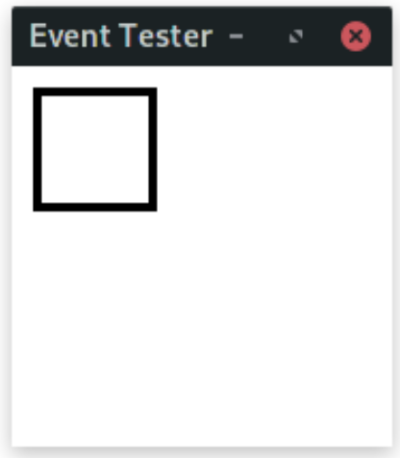


Figure 3: Event Tester Window

```
MotionNotify event, serial 37, synthetic NO, window 0x2200001,
    root 0x1e7, subw 0x0, time 17251413, (43,107), root:(1473,454),
    state 0x0, is_hint 0, same_screen YES
    state 0x0, is_hint 0, same_screen YES
MotionNotify event, serial 37, synthetic NO, window 0x2200001,
    root 0x1e7, subw 0x0, time 17253417, (143,107), root:(1573,454),
    state 0x0, is_hint 0, same_screen YES
```

Figure 4: Mouse movement test results