TESTING EMBEDDED SYSTEMS
SUBJECTS COVERED

• Unit testing (Unity)

• BDD (Cucumber) as a front-end for functional & acceptance tests

• Orchestrating tests across multiple targets
CHALLENGES TESTING EMBEDDED SOFTWARE

- Run tests locally
  - Hope that it cross-compiles

- Write simulator
  - Run tests on simulator
  - Hope that it runs on hardware

- Develop hardware
  - Run tests on hardware
  - Optimal
SOLID TESTING
void test_1 (void) 
{
    TEST_ASSERT_EQUAL(2+2, 4);
}

void test_2(void) 
{
    TEST_ASSERT_EQUAL(1+1, 3);
}
int main(void)
{
    SetupTests();
    RUN_TEST(test_1);
    RUN_TEST(test_2);
    TeardownTests();
}

BUILD UNIT TESTS
FLASH UNIT TESTS
SEE IT RUN
BDD FOR EMBEDDED
THE CASE FOR BDD

• Describes the behaviour in simple English
• Promotes collaboration within the product team
• Highlights business value
• Direct mapping from user story acceptance criteria
• Living documentation, unified view of the product
Feature: Patient monitoring

Scenario: Alert nurse on disconnect

Given patient is monitored
When I disconnect the monitor
Then I am alerted
IMPLEMENT A SIMULATOR

class Monitor
def disconnect
  driver.led(RED, ON)
end
end
Given(/^patient is monitored$/) do
  pending
end

When(/^I disconnect the monitor$/) do
  monitor.disconnect
end
Feature: Patient monitoring

Scenario: Alert nurse on disconnect
  Given patient is monitored
  When I disconnect the monitor
  Then I am alerted

1 scenarios (1 passed)
3 steps (3 passed)
0m0.0052s
WHEN SIMULATION IS NOT ENOUGH
IN-SITU TESTING

TCP/IP

Serial

Wire Server

RPC Server

Device APIs

TCP?

BLE?
THE “WIRE”

• When your system does not have native support

• When you want a lean, portable implementation
SIMPLIFIED WIRE PROTOCOL

1: begin_scenario
   1.1: success

2: step_match
   2.1: success

3: invoke

3.1: API Call
   3.1.1.1.1: Success/Fail
   3.1.1.1.1: retVal

4: end_scenario
   4.1: success
WIRE IMPLEMENTATION
BLUEPRINT

• TCP/IP loop managing Cucumber protocol

• Function table for API invocation

• API proxy implementation returning status to Cucumber
HOST HOOKING CUCUMBER TO WIRE SERVER

features/step_definitions/cucumber.wire
host: host
port: 3901
while(1) {
  getRequest(...);
  handleRequest(...);
}

Listen
int begin_callback(...) {
    serial_open(...);
}

int end_callback(...) {
    serial_close(...);
}
UP CLOSE AND PERSONAL
int patient_is_monitored(...) {
    serial_write(...,"EXEC 0\r");
    serial_read(...);
    return(retVal);
}
while (true) {
    chr = uart_read_byte();
    handle(command_buffer);
}
DEVICE API IMPLEMENTATION

```c
if(strstr(command, "1"))
{
    nrf_gpio_pin_clear(GREEN);
    nrf_gpio_pin_set(RED);
    return("0\n");
}
```
if(retVal == 0) {
    strcpy(buffer, "["success"]\n");
} else {
    sprintf(buffer, "["fail", ...);
}
RUNNING THE TEST
SEE IT RUN
REAL APPS NEED THREADS

RPC
Loop
Thread

Queue

App
thread

Implementation

Queue

Queue
IMPLEMENTATION STACK
WORKING WITH CUCUMBER

• Decide on a strategy (off-board, on-board)

• Get appropriate toolchain (cross compiler, linker)

• Implement and port Wire to target

• Run the feature files

• fail/implement/pass/refactor/repeat
SCRIPTING THE DEVICE

TCP/IP

Wire Server

RPC Server

Device APIs

Thread

Thread

Given
When
Then
COMPLEX ENVIRONMENT
GATEWAY

• Acts as an end-to-end test orchestrator

• Switchboard events across heterogeneous devices
COLLABORATIVE END-TO-END TESTING

Framework running on PC

Cucumber-Wire running on Target

Native

Targets

C-implementation

Collaboration

Wire
GATEWAY ARCHITECTURE

SpecFlow
Cucumber
Behave

Proxies
A₁
B₁

Hardware

Target A
Target B

Serial
Wire
Feature: Alarm assured to appear in quiet mode

Scenario: Pressure alarm
  Given device is in quiet mode
  When pressure sensor is disconnected
  Then a silent alarm will appear
public class QuietModeSteps
{
    SignalSimulator signalSimulator = new SignalSimulator();
    MedicalDevice medicalDevice = new MedicalDevice("192.168.1.1", 3901);

    [Given(@"device is quiet mode")]
    public void GivenDeviceIsQuietMode()
    {
        Assert.IsTrue(medicalDevice.SetQuietMode());
    }

    [When(@"pressure sensor is disconnected")]
    public void GivenPressureSensorIsDisconnected()
    {
        Assert.IsTrue(signalSimulator.SetPressure(0f));
    }
}
class MedicalDevice
{
    public MedicalDevice(string ipAddress, int port)
    {
        wire = new Wire(myAddress, port);
        wire.Open();
    }

    public bool SetQuietMode()
    {
        wire.Send("["step_matches",
                    {"name_to_match":"set quiet mode on"}]]\n");
        wire.Send("["invoke",{"id":"7","args":["on"]}]\n");
        return(wire.Ack());
    }
}
public class Wire
{
    public int Open()
    {
        client = new TcpClient(myAddress, myPort);
        stream = client.GetStream();
        return(Send("["begin_scenario"]\n");
    }

    public int Close()
    {
        stream = client.GetStream();
        Send("["end_scenario"]\n");
        return(client.Close());
    }
}
Given ... quiet mode

```
int SetQuietMode("on") {}
```

Match:

```
"set quiet\'(on|off)\"
```

Invoke:

```
idx:0, params: "on"
```

```
int set_quiet(char* state){}
```
Security - Anyone can connect to Wire!

Regulation may not allow non-application code on a production system

Shut down the wire thread in production
LESSONS LEARNED

Threads & Target Architecture

Vocabulary
OPEN SOURCE

• Unit testing example
  https://github.com/ihassin/nrf51-unity

• Cucumber/Listener/RPC example
  https://github.com/ihassin/cucumber-wire-tcp2serial

• Development environment provisioning (Linux)
  https://github.com/ihassin/fruitymesh-ubuntu-vm

• Development environment provisioning (OS-X)
  https://github.com/ihassin/fruitymesh-mac-osx
REFERENCES

- Unity
- Cucumber
- Specification by example
- The Cucumber Book
- Cucumber Recipes
- SpecFlow
- Nordic Semiconductor

Photo Credits:
@history_pics/@historyinpics
Jim Reese#Wikipedia
National Library of Australia
THANK YOU!

@itababy
www.in-context.com