The Aston University Internet of Things (IoT) application trainer manual

Introduction

The board (refer to Figure 1) is based around an Adafruit ESP8266 module. It is capable of implementing numerous IoT applications via the inclusion of:

- A DHT22 digital temperature and humidity sensor.
- Analogue input: A potentiometer or a light dependent resistor (LDR) light measurement circuit can be connected to the ESP8266's single analogue to digital converter (ADC) via the switch placed below the A0 signifier.
- A VMA405 MFRC522 based RFID card reader module. This is capable of **reading** and **writing** of **Mifare** cards operating at 13.56 MHz.
- A 16x2 character liquid crystal display (LCD) with a programmable RGB backlight.
- A momentary push to make button and an indicator LED. Note that both the button and the LED share the same IO pin. Hence, the button and the LED cannot be used at the same time. Refer to the guidance below for further details.
- A user Inter-Integrated Circuit (I^2C) bus Grove connector for user expansion.
- A Lithium Polymer (Li-Po) charge and power circuit. Note that the boards VCC supply will nominally be 3.7V under battery power.

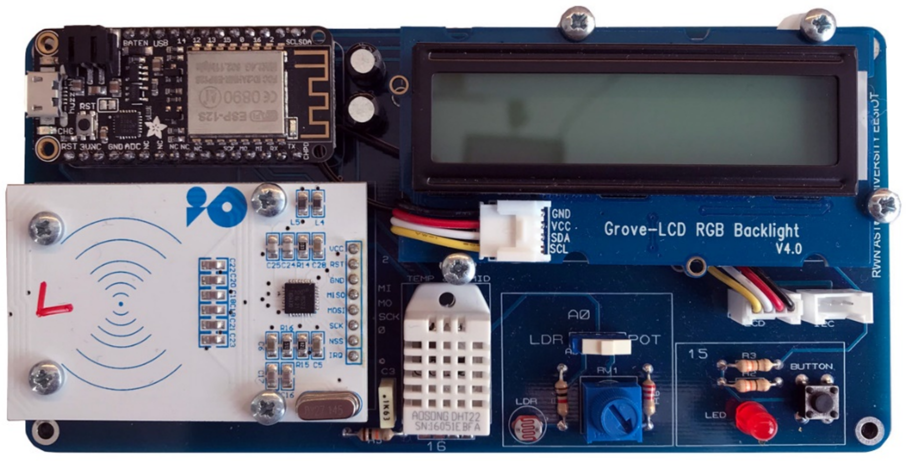


Figure 1: The Aston IOT hardware platform

The board's layout and schematic is given in Figure 8 and Figure 7 respectively. A pin-mapping table is provided in Table 7.

Temperature and Humidity sensing - DHT22

A digital temperature and humidity sensor is included such that the Aston IoT application trainer can be utilised as a wireless sensor mote. The DHT22 is a device which contains a ceramic humidity sensor, a thermistor for temperature measurement and an embedded system to sample the sensors and communicate the data to a host system.

The communication protocol makes use of a single IO pin in which the host or the DHT sensor can only pull the line low (similar to I^2C). A 10kΩ resistor is used to pull the line high. Data is encoded via the length of pulses generated by the DHT22.

Table 1: Pins utilised for the DHT22 1 wire communication protocol

Pin	**Comments**
16	Communicates to the DHT via a single wire communication protocol. A 10kΩ resistor pulls the line high.

Analogue input

The ESP8266 module contains a single channel 10-bit ADC, denoted as **A0** in the Arduino IDE. Hence, if multiple analogue devices are to be connected, an external analogue multiplexor must be used. In the case of the Aston IOT application trainer, a simple single-pole double-throw (SPDT) switch is used for this purpose to select between the potentiometer and the LDR light measurement circuit. However, digitally controlled analogue multiplexors (such as the 4051) could be used in ESP8266 based designs to expand the number of ADC channels and for the ESP8266 to have control over which analogue input it is sampling.

It is worth noting that the ADC only accepts voltages in the range of 0 to ~1V. Hence, the range of ADC values is:

$$0\text{V} = 0$$

$$\sim 1\text{V} = 2^{10} - 1 = 1023.$$

Giving a resolution of $1\text{V} \div 2^{10} = \sim 977\ \mu\text{V}$.

Potential dividers are utilised to scale the voltage from the potentiometer and LDR to within the 0 to 1V useable range.

Table 2: Pins relating to the ADC

Pin	**Comments**
A0	The only ADC analogue input pin available on the ESP8266. Input impedance is estimated to be approximately 1MΩ.

RFID module – VMA405

A VMA405 / MFRC522 radio frequency identification (RFID) module is provided for the reading and writing of Mifare classic contactless smart cards, with two types being shown in Figure 2.

Figure 2: Mifare RFID tag and card

The interface between the module and the ESP8266 is a serial peripheral interface (**SPI**) bus, which can be thought of as four shift registers. The microcontroller transfers the contents of its transmitting shift register to the MFRC522's receiving shift register whilst the MFRC522 transfers the contents of its transmitting shift register to the microcontroller's receiving shift register at the same time. This enables high speed **full-duplex** serial communication.

Table 3: Pins utilised for the RFID module

Pin	**Comments**
SCK	SPI Clock
MO	SPI serial data. Master Out Slave In.
MI	SPI serial data. Master In Slave Out.
2	A reset/user IO pin (typically used to reset devices connected via SPI)
0	SPI chip select (CS).

Button and LED

Due to the limited amount of input/output (IO) pins on ESP8266, the indicator LED and the push-to-make button share the same IO pin, as shown in Figure 3. Note that some IO pins on the ESP8266 are actually duplicates of other pins. Hence, the "unused" IO pins present in the design.

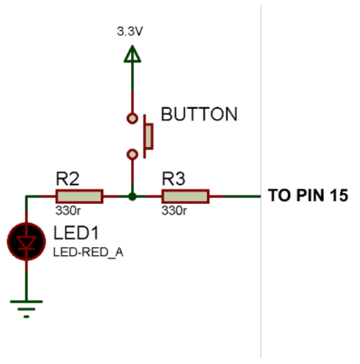


Figure 3: Schematic snippet demonstrating how the LED and button share the same digital IO pin (15). ***Note that pin 15 is pulled down on the ESP8266 module.***

R3 and **R2** limit the current drawn by the LED when the IO pin is driving it and **R2** limits the current for the LED when the button is pressed. **R3** also serves the purpose of protecting the IO pin in the event that:

a. Pin 15 is set as an output,
b. Set LOW (i.e. approximately 0V),
c. The button is pressed.

Otherwise, the IO pin would be damaged from excessive current as the pin would be directly connected to 3.3V.

Using the LED:

The LED can be used normally by setting the pin as an output with **pinMode** and writing to the IO pin via **digitalWrite**.

Reading the button:

Reading the state of the button is a little more complex. The approach taken is to repeatedly call a C function that:

1. Sets pin 15 as an **input**.
2. Reads the current state of the button and checks to see if a rising edge has been detected (i.e. a logic 1 has just been sampled and the previous sample is a logic 0).
3. Perform another check of the buttons state after approximately **5ms** to ensure that:
    a. We are detecting the rising edge of an actual button press and not false transitions caused by switch bounce.
    b. We are detecting the switch being pressed and not released. Switch bounce also occurs on the **release** of a switch/button.

    Figure 4 demonstrates how this approach works and why the second check of button state after the bounce period is necessary.

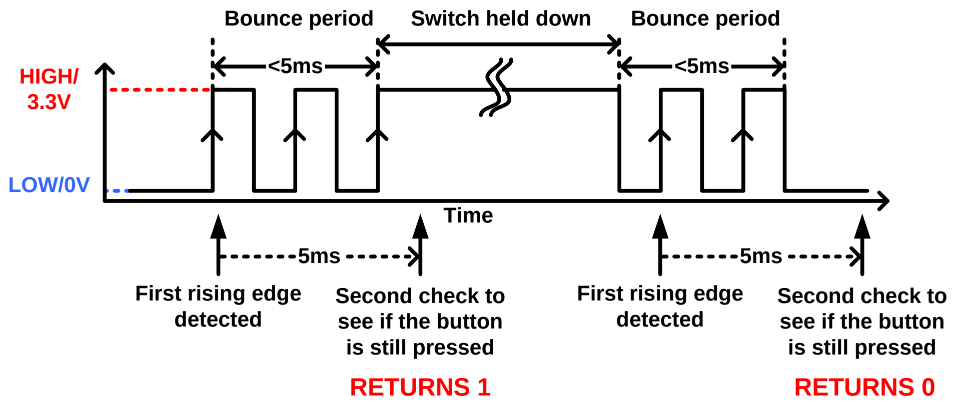

Bounce period
Switch held down
Bounce period
<5ms
<5ms
HIGH/ 3.3V
LOW/0V
Time
5ms
5ms
First rising edge detected
Second check to see if the button is still pressed
RETURNS 1
First rising edge detected
Second check to see if the button is still pressed
RETURNS 0

Figure 4: Switch bounce and the algorithms sampling points

*Note that **5ms** is an empirically found figure for the switches used on the Aston IoT application trainer boards. The length of switch bounce can vary according to the switch type and manufacturer – it is good practice to measure it and then wait for a time that exceeds the longest bounce measured.*

4. Set pin 15 back to an output and update the last sampled variable with the current button state.
5. If the input value is still a logic 1, return a 1 to indicate to the calling function that a button has been pressed. Otherwise, return a 0.

This process is shown diagrammatically as flow chart in Figure 4.

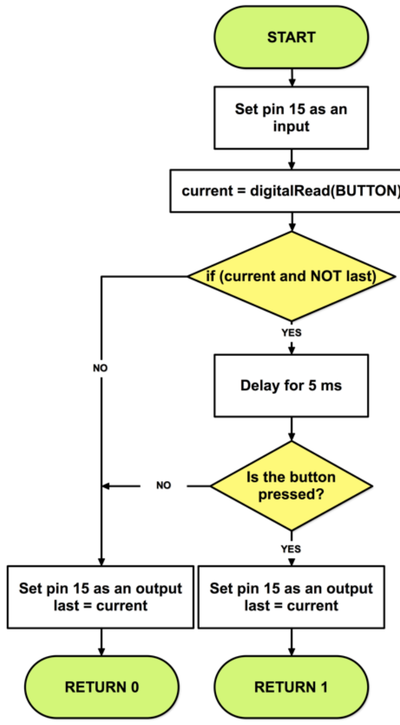

START
Set pin 15 as an input
current = digitalRead(BUTTON)
if (current and NOT last)
YES
NO
Delay for 5 ms
Is the button pressed?
NO
YES
Set pin 15 as an output last = current
Set pin 15 as an output last = current
RETURN 0
RETURN 1

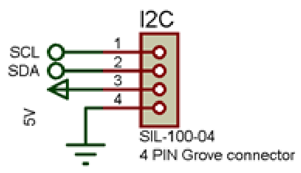
Figure 5: A simple switch debounce algorithm to detect a button press

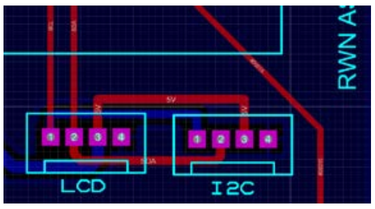
Table 4: Pins utilised for button and LED IO

Pin	**Comments**
15	A single user IO pin that is connected to an indicator LED and a push-to-make tactile button. **Note that the ESP8266 module contains a pull-down resistor.**

I^{2}C

An I^{2}C bus is utilised to communicate with a Grove RGB LCD shield, which is capable of displaying alphanumeric characters and changing the colour and brightness of its backlight. Note that typically, pull-up resistors are required for I^{2}C. However, the internal pull-ups are utilised on the ESP8266 module to simplify the design. Figure 6 shows the Grove connector pin-out and an additional 4-pin grove connector marked as **I2C** is provided for user expansion. I^{2}C is significantly slower than SPI due to the reliance on resistive pull-ups. This introduces an RC time constant limit to the maximum potential bit rate. However, there is no requirement of a chip select (CS) for each device connected to the bus.





(a) *(b)*
Figure 6: Schematic (a) and PCB layout (b) for the Grove I2C connector

Table 5: Pins utilised for I2C communication

Pin	Comments
SCL	I2C serial clock. Connects to the Grove RGB LCD module and the I2C expansion header.
SDA	I2C serial data. Connects to the Grove RGB LCD module and the I2C expansion header.

Pin mapping

Table 6: The platform's pin mapping

Pin	Connected to	Comments
15	LED and user button.	Refer to notes above on use of the LED and button.
16	DHT humidity and temperature sensor.	Single wire communication protocol. A 10kΩ resistor pulls the line high.
SCL	Grove RGB LCD module and the I2C expansion header.	I2C serial clock.
SDA	Grove RGB LCD module and the I2C expansion header.	I2C serial data.
A0	Either the LDR or potentiometer circuit.	The only ADC analogue input pin available on the ESP8266. Input impedance is estimated to be approximately 1MΩ.
SCK	VMA405 SCK	SPI serial clock
MO	VMA405 MOSI	SPI serial data. Master Out Slave In.
MI	VMA405 MISO	SPI serial data. Master In Slave Out.
2	VMA405 RST	A reset control line for the RFID module.
0	VMA405 NSS	SPI Chip Select (CS).

Module schematic

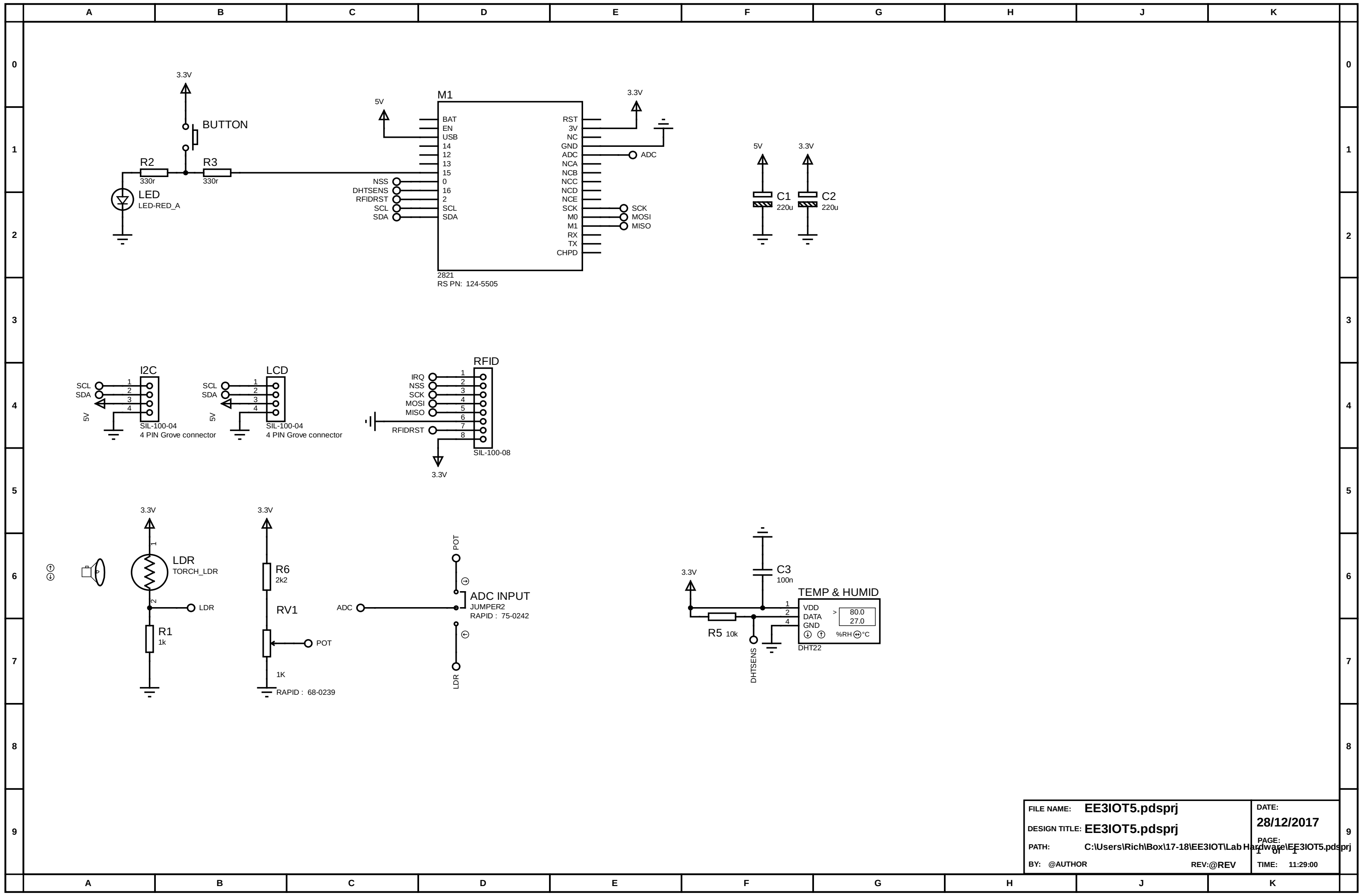


Figure 7: The schematic of the Aston IOT application trainer

PCB layout

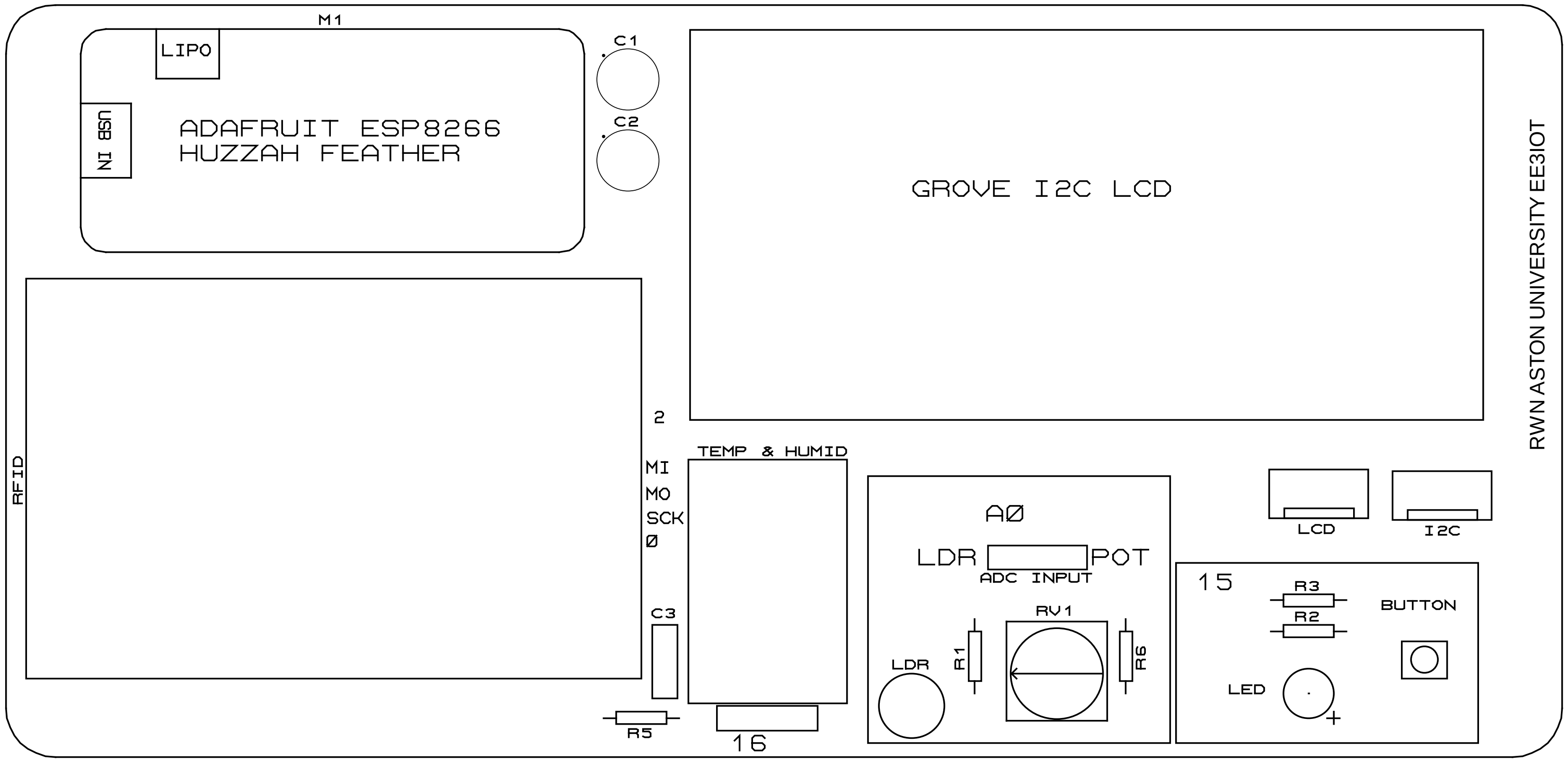


Figure 8: Layout of the PCB