The Internet of Things

Technical Overview

Doug Merrett
Lawrence Ratcliffe

Contents

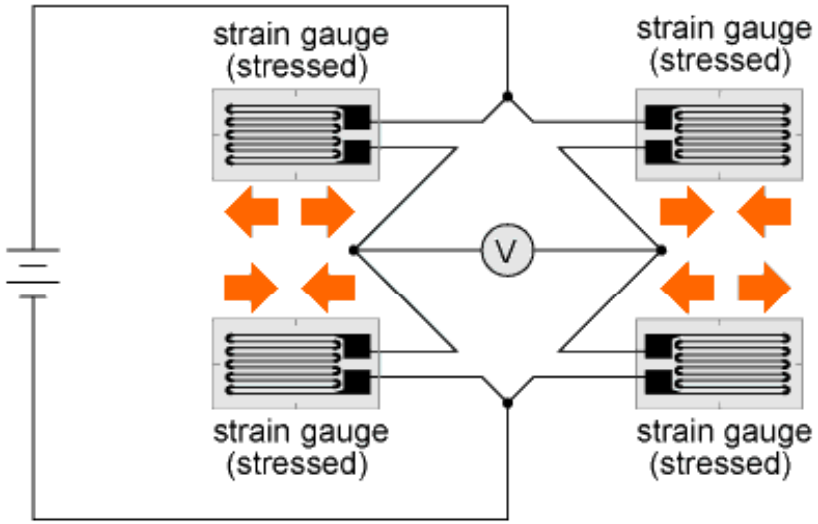
- Hardware
  - Scales
  - Printer interface
  - LCD interface
  - Arduino connections
- Software
  - Arduino
  - Java Proxy
  - Salesforce

Hardware

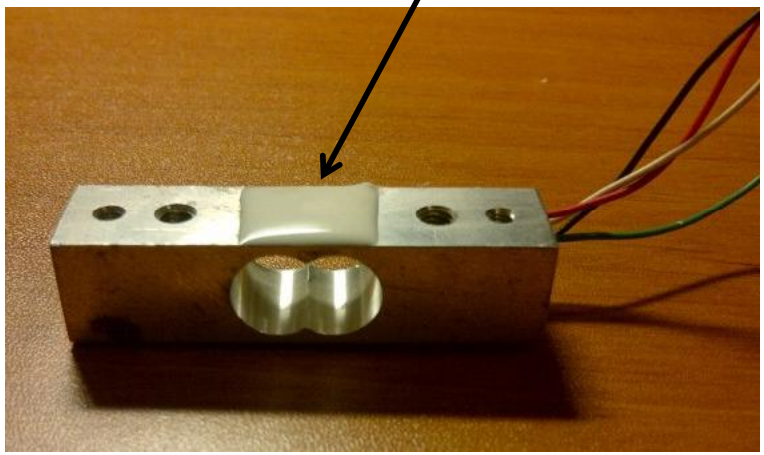
Scales

Scales – Load Cell Details

Full-bridge strain gauge circuit

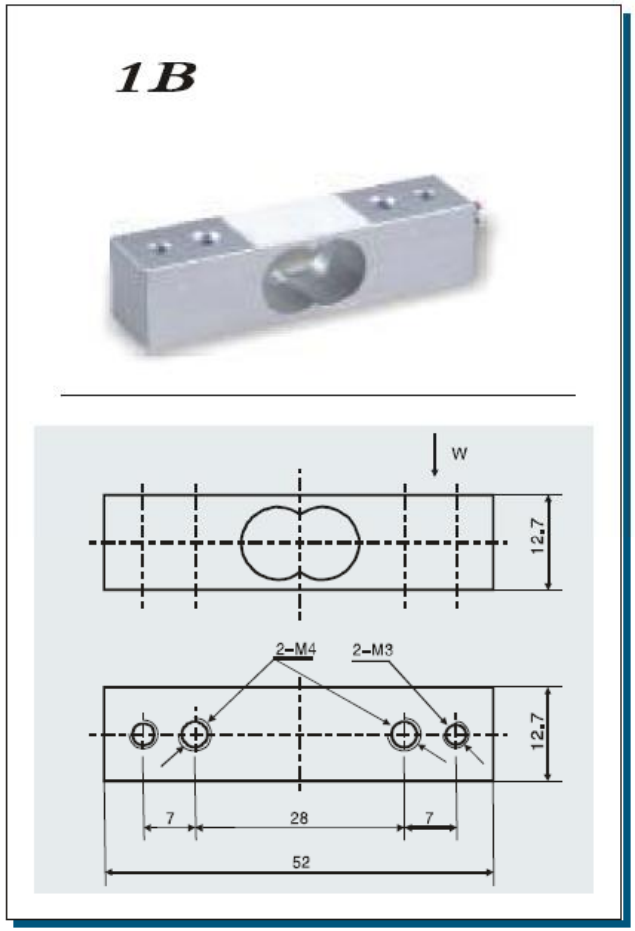


Two strain gauges on top and another two on the bottom (under the epoxy)



When the load cell flexes downwards, the strain gauges alter their resistance and the voltage difference between the midpoints grows larger. This is then amplified to make it large enough to sense with the Arduino's AD converter.

Scales – Load Cell Specifications

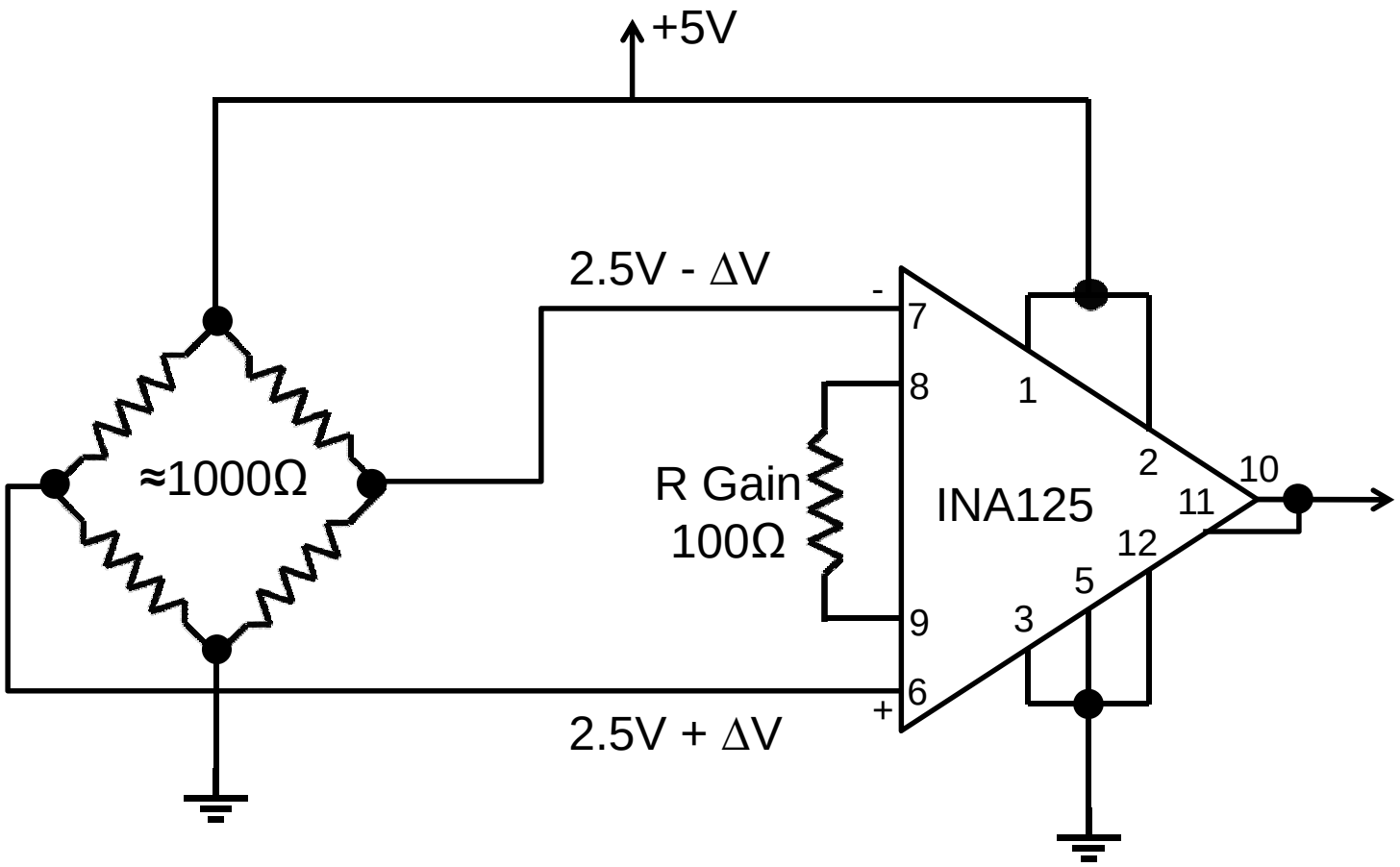


Model	1B
Capacity	5Kg、6Kg、10Kg、12Kg、15Kg、20Kg
Combined Error	0.03
Output Sensitivity	1.8±0.1 mV/V
Non-linearity	±0.02%F.S.
Repeatability	±0.02%F.S.
Hysteresis	±0.03%F.S.
Creep	±0.03%F.S.5min
Temp. Effect on Zero	±0.3%F.S./10℃
Temp. Effect on Span	±0.03%F.S./10℃
Zero Balance	±0.1 mV/V
Input Resistance	1055±10Ω
Output Resistance	1000±10Ω
Insulation Resistance	>3000 MΩ (50V)
Excitation Voltage	≤10 V
Operation Temp. Range	-10~+40℃
Overload Capability	150%F.S.
Recommended Application	Portable Scales

http://zemic.nl/Products-Single-Point-load-cells/miniature-sensors.html

I bought the 20Kg version

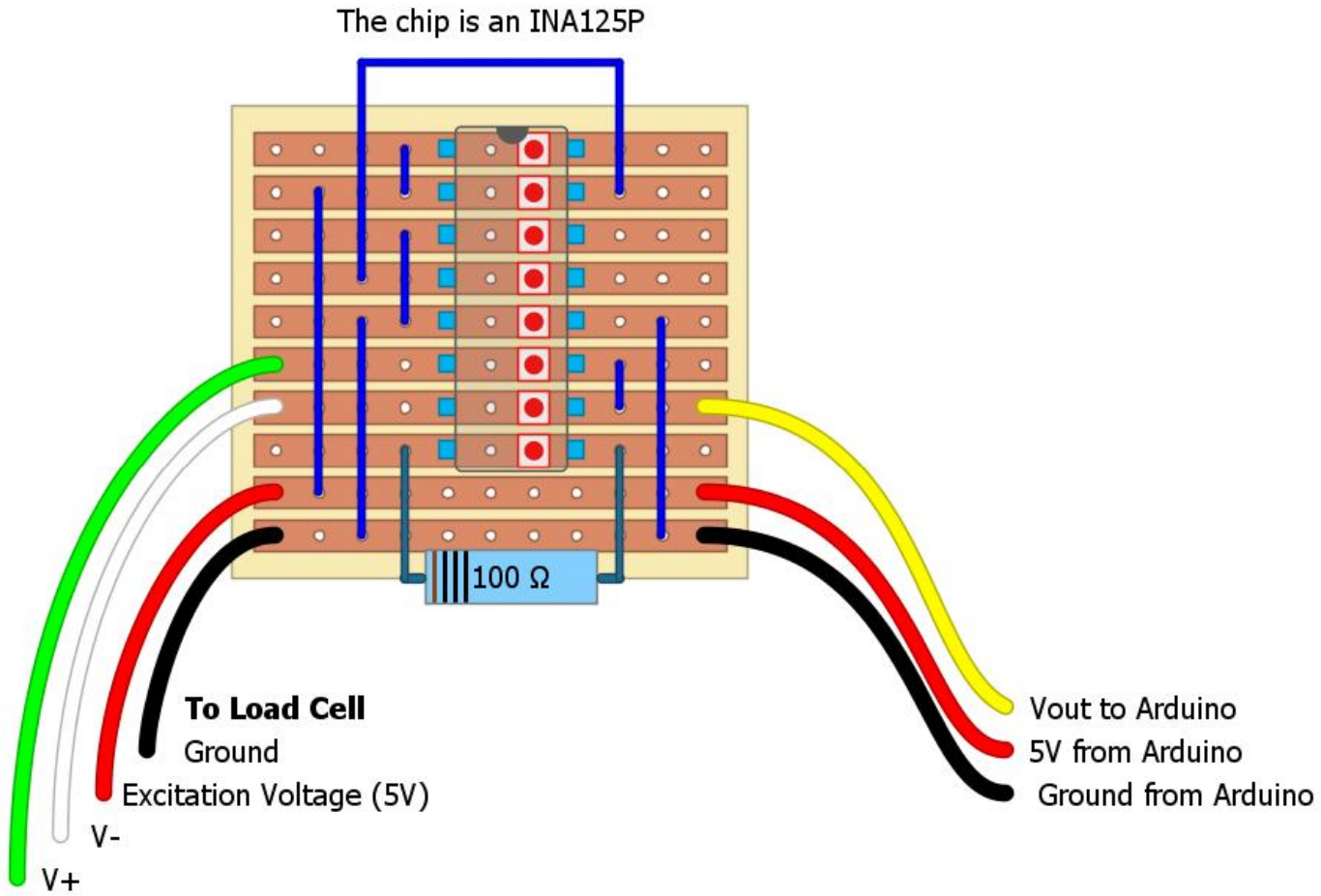
Scales – Electronic Build



Gain is set at: 4 + 60kΩ / R Gain = 604

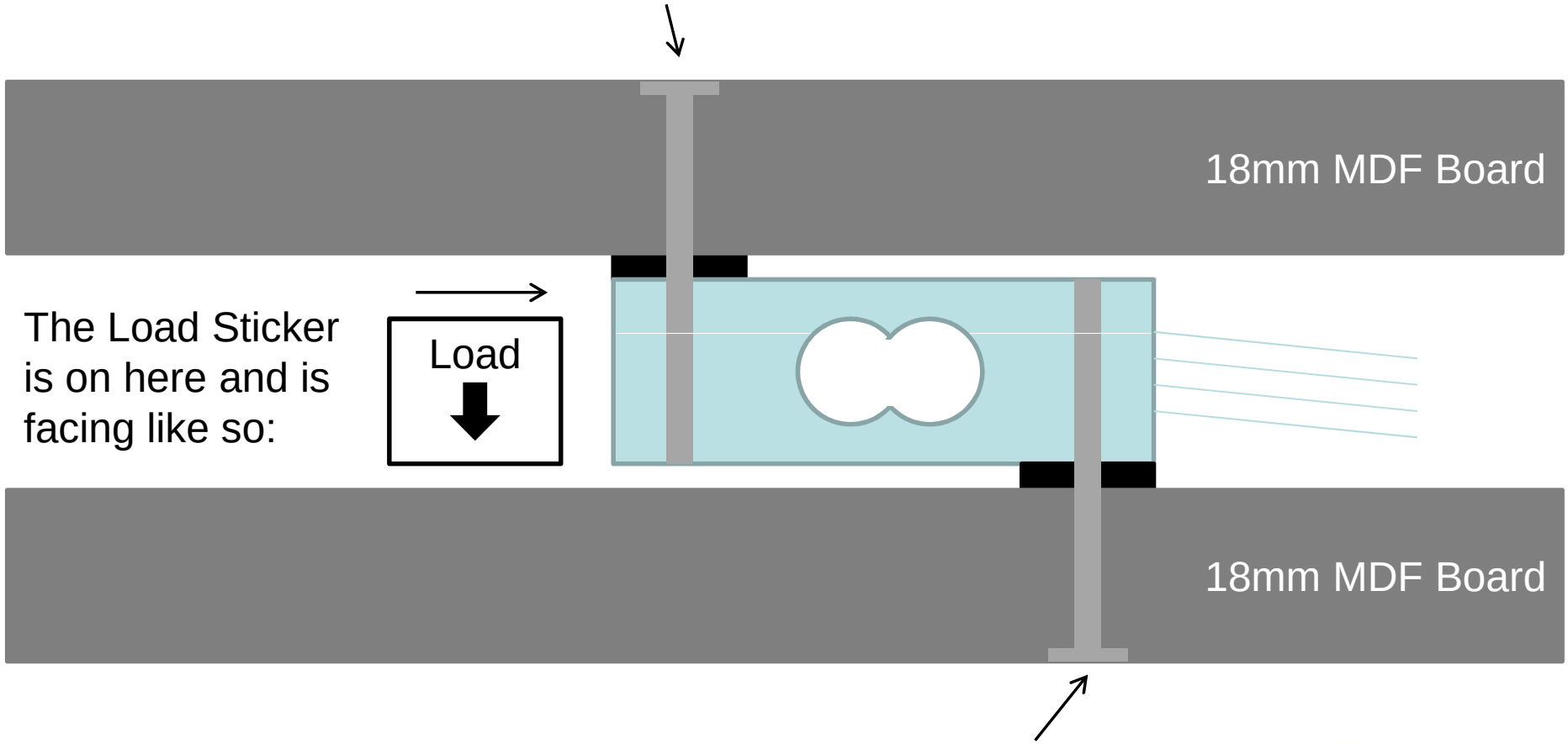
Datasheet is available at: http://www.ti.com/lit/gpn/ina125

Scales – PerfBoard layout



Scales – Physical Build

Load is applied to the top board held in place with an M4 bolt screwed into the load cell, with a 3mm washer creating the required gap



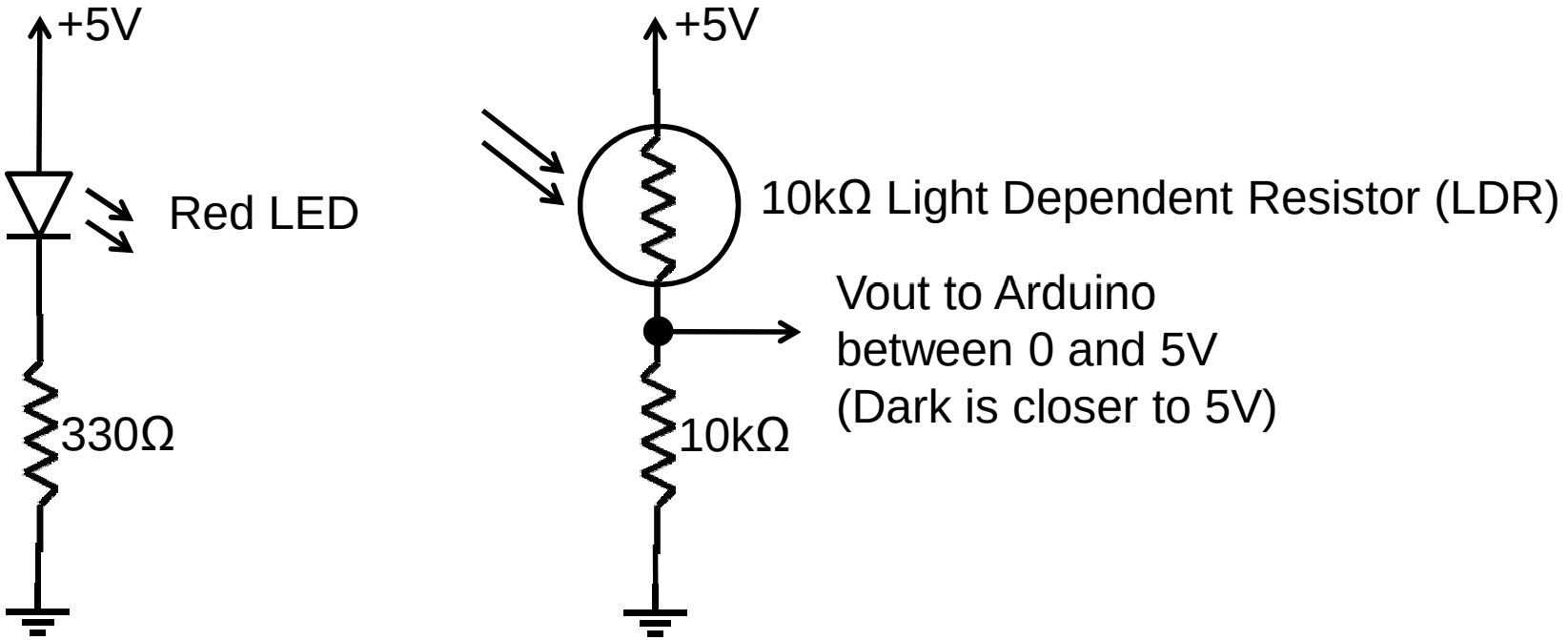
The Load Sticker is on here and is facing like so:

M4 bolt holding the load cell off the base with a washer 3mm thick

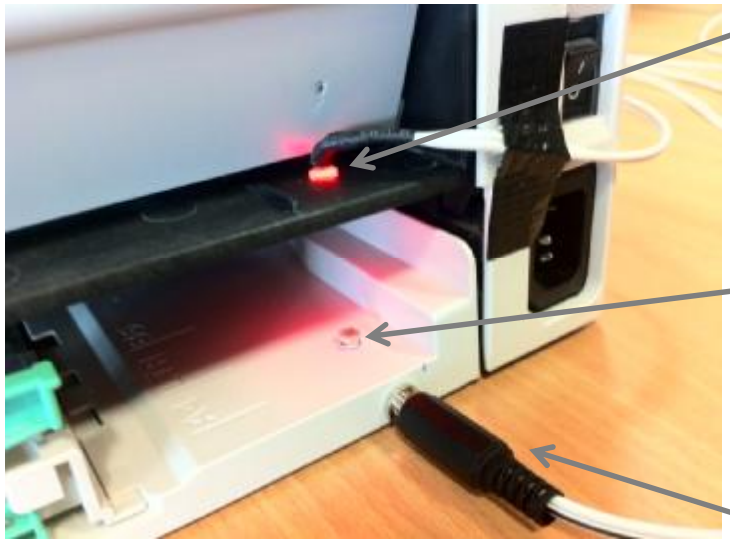
Hardware

Printer Interface

Printer – Electronic Build



Printer – Physical Build



Light Emitting Diode (LED)

Light Dependent Resistor (LDR)

Connector from the LDR to Arduino
(saves the cable being dragged when changing paper)

Hardware

LCD Interface

LCD – Electronic Build

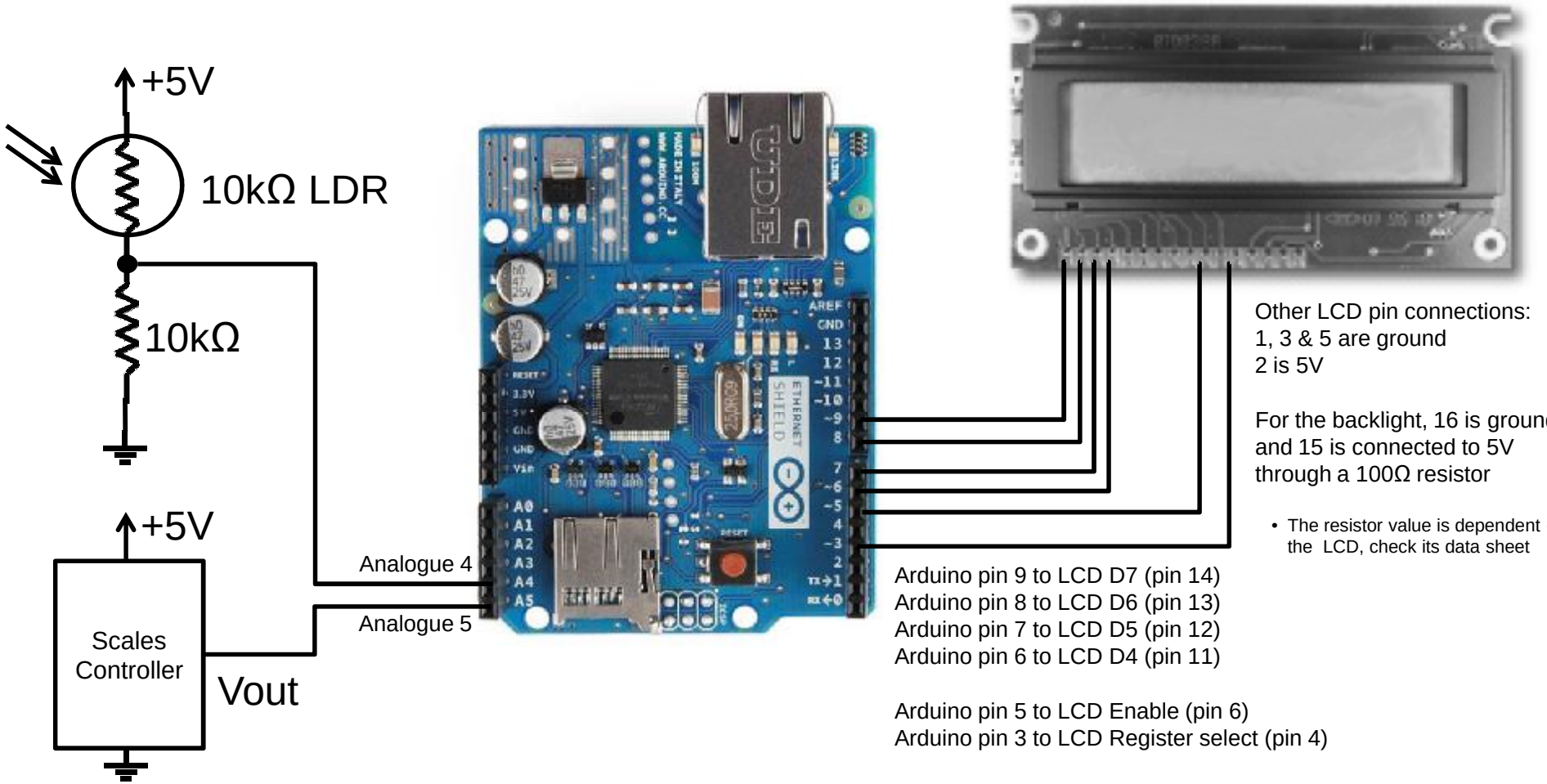
16 Character x 2 Line LCD
(with LED backlight)

16 pin header soldered to the LCD module to allow it to be pushed into the breadboard

Datasheet is available at: http://www.powertipusa.com/pdf/pc1602f.pdf

Arduino Connections

Arduino Connections



The reason for “non-standard” LCD pins is that the Ethernet shield uses Arduino pins 2, 4 and 10 through 13

Software

Arduino

Arduino Software

- I used Arduino V1.0 for the build
- The libraries I used were the inbuilt Ethernet and LCD, the SDFat library from http://code.google.com/p/sdfatlib/ as it was easier to interface with and provided “C++ stream functions” for parsing the config file
- All configuration information is loaded from a config file loaded from the SD Card on the Ethernet Shield
- The LCD is used for status and error messages
- The Serial output is utilised for detailed logging

Arduino Software (cont)

- The configuration file on the SD card is in this format

```
// CONFIG FILE FOR ARDUINO - INTERNET OF THINGS
// Doug Merrett - Salesforce.com 2011

// The MAC address of the Arduino
0x90 0xA2 0xDA 0x00 0x88 0x6B

// Arduino IP Address
192.168.5.50

// Proxy IP Address
192.168.5.22

// Proxy port
4444

// The name of the device
LS.F0.CLO

// Is it connected to a Scales (S), Printer (P) or Both (B)
B

// The ANALOGUE input pin for the weight (must exist, but is ignored for Printer only)
5

// The ANALOGUE input pin for the paper out (must exist, but is ignored for Weight only)
3
```

Arduino Software – Future Enhancements

- When the 0x7000 byte Sketch limit bug is fixed and there is more program space
  - Use DHCP for the Arduino IP address
  - Use the SD Card for configuration information if it is there, and then save the configuration into the EEPROM
  - If the SD Card is not installed, use the EEPROM to get the configuration information

Software

Java Proxy

Java Proxy

- Because the Arduino can’t do HTTPS connectivity to allow for direct Salesforce.com integration, I built a Java Proxy service that runs on the local network to allow for all external communications to be encrypted.
- The Proxy is multi threaded and will support many Arduino devices on one network
- It is written to take a simple string on a socket and then call the Salesforce SOAP API to update the appropriate Printer or Store record

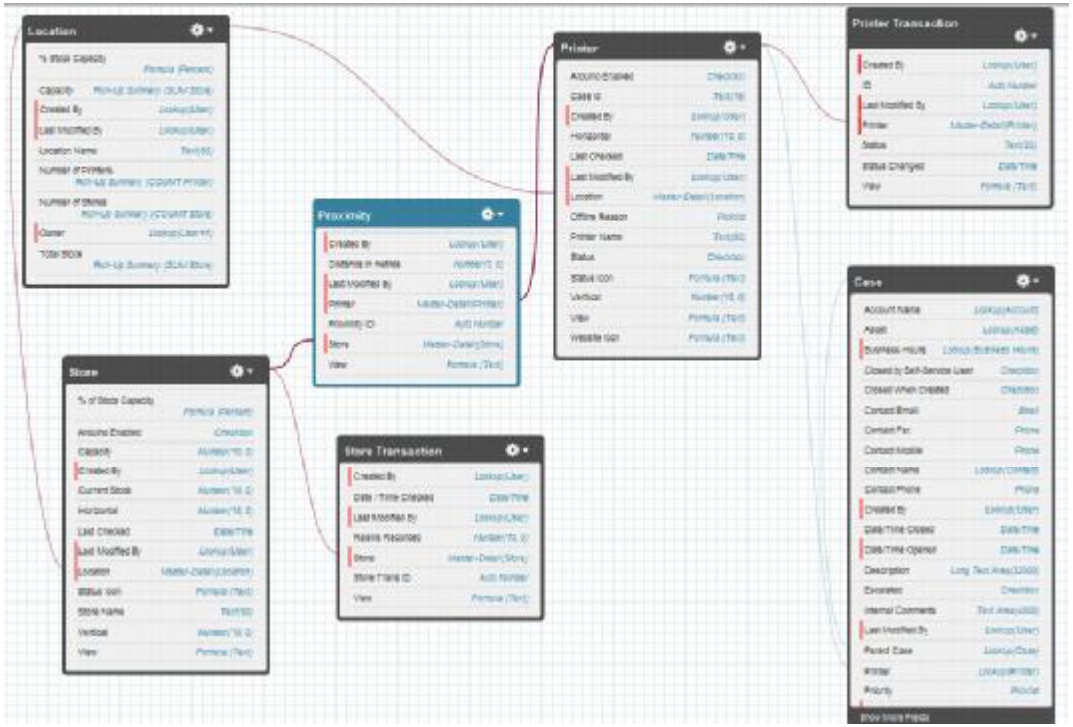
Java Proxy

- The proxy uses the Salesforce WSC library for the SOAP API and is extremely easy to use (http://wiki.developerforce.com/page/Introduction_to_the_Force.com_Web_Services_Connector)
- The simple string is in this format:
  - P,PrinterName,X X is an integer, 0 = paper out, 1 = OK
  - S,StoreName,X X is the number of reams of paper
- All configuration is by command line parameters
  - java ArduinoProxy port user pass
    - port - an integer between 1024 and 65535
    - user - the Salesforce username (testuser@company.com)
    - pass - the Salesforce password (followed by optional security token)

Software

Salesforce

Force.com Data Model



6 custom objects were created on Force.com to represent printers, paper stores etc. The standard “Case” object was used, renamed as “Service Request”.

Force.com Data Model (cont)

- Location
  - An office location and represents a summary of all the various paper Stores
- Store
  - An area where reams of paper are stored
  - The Arduino is monitoring the scales in each store and populating the current number of reams in that store
- Store Transaction
  - Used for historic purposes to track the changes happening within the Store object

Force.com Data Model (cont)

- Printer
  - A physical printer representing its current status
- Printer Transaction
  - Used for historic purposes to track the changes happening within the Printer object
- Proximity
  - Junction object that joins Stores and Printers together and their distance apart
- Case
  - Standard object, renamed to “Service Request”
  - Automatically opened and closed via Apex based on different events

Apex

- The only Apex used are three triggers to cope with the updates caused by the Java Proxy responding to the Arduino inputs. All are “bulkified”.
- BeforeUpdateStore trigger
  - Set the number of reams of paper in a store
  - Create a Store Transaction record for this update
  - Makes sure that there is one transaction per day for each Store in the Daily Store History table used for dashboards and reports
  - If the number of reams falls below 3 in a store, then raise a case to provide for pre-emptive restocking

Apex (cont)

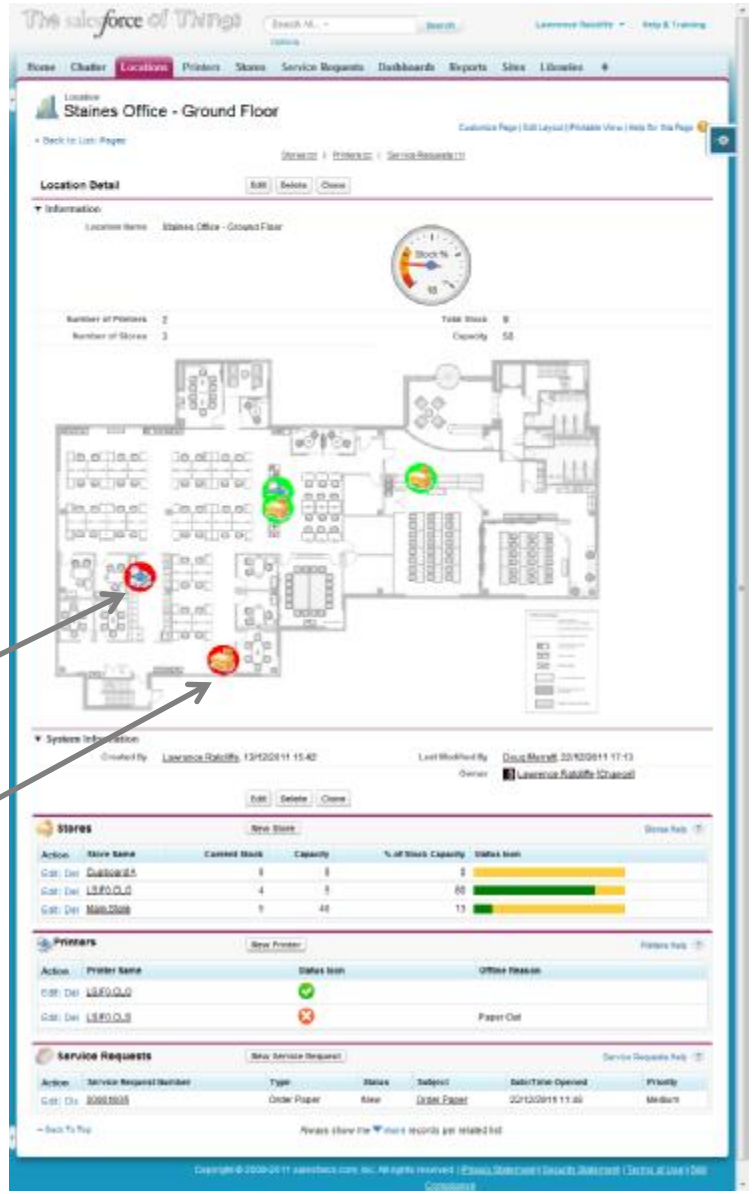
- BeforeUpdatePrinter trigger
  - Set the printer's status
  - Create a Printer Transaction record for this update
  - If status is Paper Out (from Arduino), then raise a case saying the paper is out and tell the agent where the closest stores with paper are situated in the case description
  - If there is an existing Paper Out case, and the printer has gone Online again, automatically close the case

Apex (cont)

- BeforeUpdateLocation trigger
  - This is triggered by the RollUpSummary field holding the total number of reams of paper
  - If there are five or fewer reams of paper in the whole location, then raise a case to order more paper
  - To stop too many cases being raised, only create a case if there are no currently open “order paper” cases or the closed date of the case is more than 5 days ago

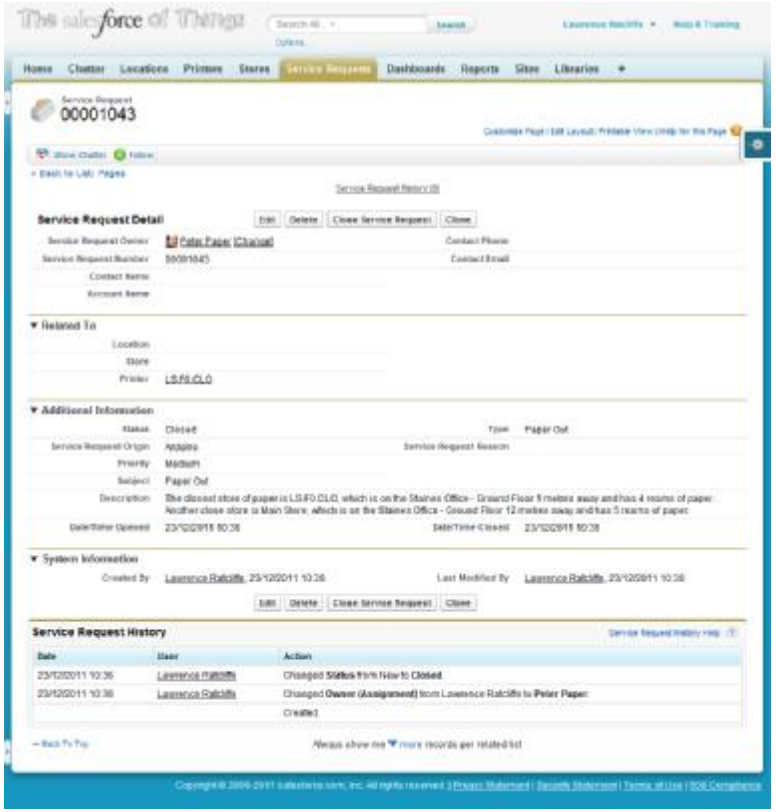
Visualforce

- Visualforce has been used to render inline the real-time location and status of Stores and Printers



Service Requests (Cases)

- A Service Request is automatically created when paper out is detected by the Arduino
- The Description tells the Service Request Owner where the closest available paper stock is based on current stock and proximity

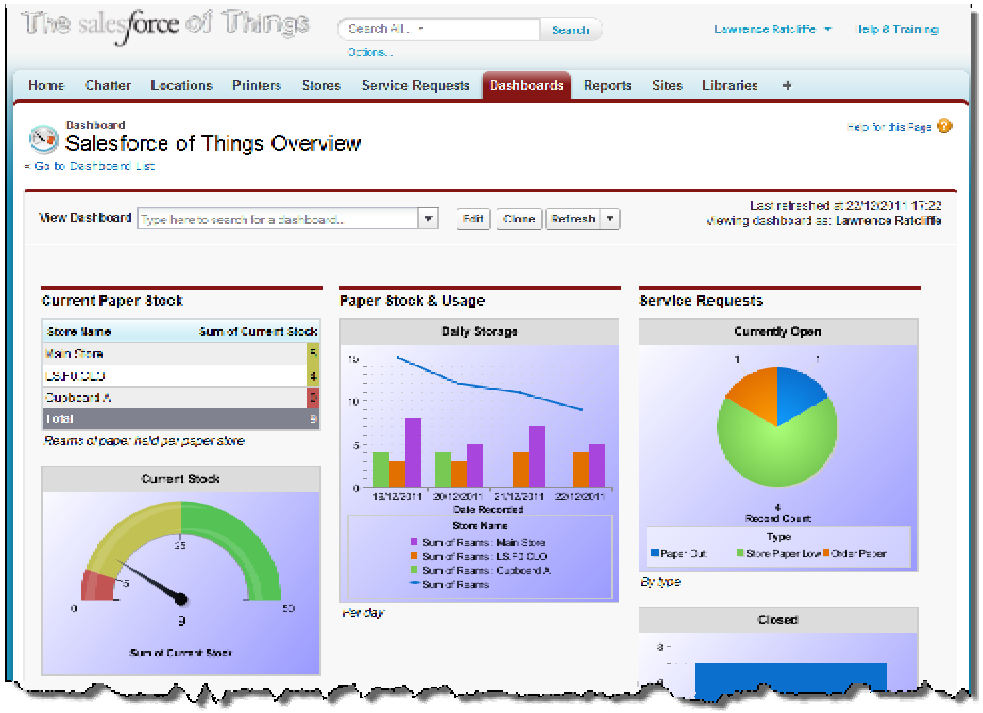


Siteforce

- A website has been built on Siteforce to summarise the project and provide the real-time status of the Printer and Store for the “mobile app”
- http://bit.ly/tvW7W5 (via Firefox or Chrome)

Dashboard

- A set of reports and a dashboard have been configured to visualise the paper usage and Service Request (Case) volume history



Thank you!