**Notes for Report**

* Using pro mini dev board over Arduino Uno as it is physically smaller cheaper and runs on 3.3v whereas the Uno runs on 5v cons that it has a fixed clock speed of 8MHz however for my prototype uses this is not an issue
  + Can be directly soldered too as to reduce the form factor of the probe
  + Direct soldering means I can determine the length of cable the mini has from the pi
* DS18B20 bought for £2.45 on ebay.co.uk
* Pro Mini 8MHz 3.3v / 5v MEGA328P bought on ebay.co.uk for £1.50
  + Fixed to 8MHz has variable 3.3v / 5v input supply
  + Tx and Rx used for serial connection to Raspberry Pi
* Soldered 6 pin header to the Pro Mini for ease of programming and data transfer
* Soldered the DS18B20 to the A3 VCC and GND pins of the Pro Mini connecting the VCC and Data lines via a 4kΩ resistor in the shape of 2 2KΩ resistors connected in series. (due to lack of 4kΩ resistor), (made the mistake of not adding this at first and go bad readings)
* Wrote several attempts at the programming of the pro mini with multiple failures and multiple redesigns such as implementation of state machine and removal of the delay function and the addition of the millis() function to act as the basis of a timer and interrupt sequence.
* Set up the raspberry pi:
  + Ran following update commands:
    - sudo apt-get update
    - sudo apt-get dist-upgrade -y
  + Ran the following uninstall commands
    - sudo apt-get purge minecraft-pi wolfram-engine scratch -y
    - sudo apt-get scratch2 libreeoffice\*
    - sudo apt-get autoremove
  + Ran the following install commands:
    - sudo apt-get install apache2
    - sudo apt-get install motion
      * sudo nano /etc/modules
        + adding “bcm2835-v4l2” to the end of the file
      * sudo nano /etc/motion/motion.conf
        + changing daemon off to daemon on
        + changing the height and width properties to match the cameras height 768 width 1024
        + changing the framerate from 1 to 60
        + changing output\_pictures on to output\_pictures off
        + changing stream\_port 0 to stream\_port 8081
        + changing stream\_localhost off to stream\_localhost on
        + changing webcontrol\_port 0 to webcontrol\_port 8080
* Now that the web services are setup I opened Geany, one of the raspberry pi’s IDEs, to write the code for the website in two files and index.html file and a MainPage.css file that after coding were moved to the /var/www/html folder where the test index.html was replaced with my webpage’s index.html currently the page and stream are accessible on a local network only with the webpage on port 80 hosting the stream in an iframe.
* Connecting the Arduino to the raspberry pi via a UART connection took a little trial and error. First I connected the raspberry pi’s 3.3v supply (gpio pin 1) ground (gpio pin 6) TxD (gpio pin 8) and RxD (gpio pin 10) to a breadboard for easy prototyping of circuitry initially and incorrectly I connected Tx and Rx directly to the respective pins on the Arduino however at closer inspection the Tx of the Ardiuno needed to be connected to the Rx of the raspberry pi and the Rx of the Arduino to the Tx of the raspberry pi. Following this I went into the raspi-config menu with the command sudo raspi-config and went to option 5 interfacing options then to option P6 serial and disabled console over serial option but kept the hardware enabled. After setting up the required hardware options I wrote some code (adapted from instructables) and tested it, where it failed, I then ran the command ls –l /dev to see what port I was using and the name of that port in this case I was using serial0 which is /dev/ttyS0 instead of /dev/ttyAMA0 which was on the instructables article the code for getting my temperature readings from the Ardiuno into the pi now worked.
* Installing node.js for raspberry pi by running the upgrade commands sudo apt-get update && sudo apt-get dist-upgrade -y then running sudo apt-get install -y nodejs
  + running node -v verifies the version and that the install went according to plan
  + in order to use the node.js with the pi’s gpio we need to install the onoff module this was done with the npm install onoff command
  + in order to host a webservice that we can interface our AJAX and serial port javascript to we need to install socket.io this is done with the command nmp install socket.io –save
  + (follow on with code to create websever use ajax and node.js to create and updateable webpage that displays the data.)
* Wrote proof of concept code in node.js using the serialport module this code takes the data from the serial port and writes it to the console.
* Uninstalled apache2 with the command sudo apt-get purge apache2 -y
* Rebuilt the webserver using node.js it takes very few lines of code the new webserver sits on the same port at the old one (8080).
* Struggled greatly with node.js webserver realised I needed to install further libraries (express)
  + sudo npm install express