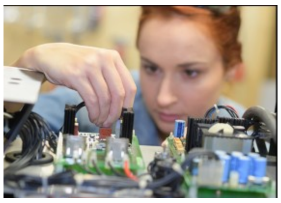
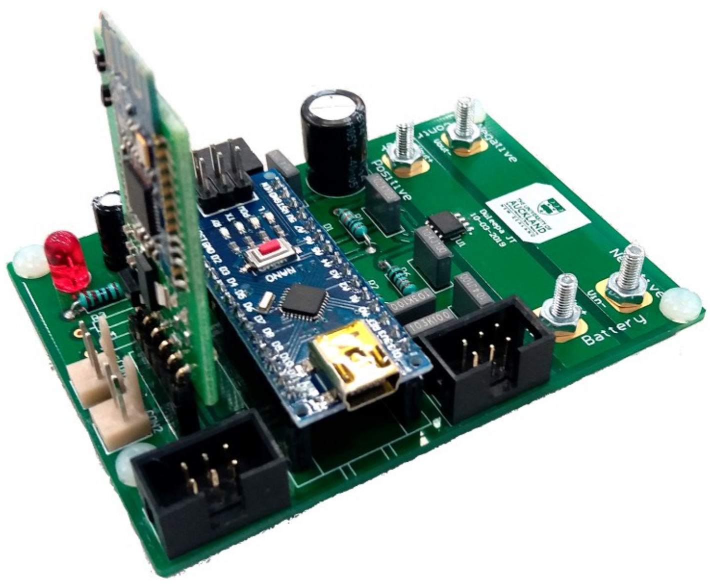


Evolocity Innovation Category :

"Measurement & Transmission of Performance Metrics"



Booklet contents

Duleepa Thrimawithana (UoA)

Les Black (Evolocity)

Peter Wilson (ARA)

For Evolocity 2019

Evolocity & Auckland University

EVolocity teams are most grateful to the Auckland University Department of Electrical, Computer & Software Engineering for the design and supply of these kits for the Innovation category.

Introduction to Microcontrollers with Arduino

Kit Contents

Check that you have these parts in your kit

	Arduino Nano	
	Auckland University breakout board	
	Solderless breadboard	
	USB cable	
	Resistor 220 ohm	
	Resistor 10K Ohm	
	10 Jumper wires in ghastly colours	
	PCB supports and screw sets	

Red and green LED	
Push switches	
9V battery and connector	
Capacitor 0.1UF,	
Temperature sensor DS18B20	
Current sensor ACS730	
Bluetooth module	

What is Arduino?

Arduino is an Open Source platform comprised of two parts the Arduino board (hardware) and the Arduino IDE (Software). IDE is the 'Integrated Development Environment' used to program the Arduino.

You will write the program in the IDE and download it to the board to execute it. The program is then embedded in the chip on the board and can operate as a completely standalone device. It is in fact a tiny computer you can program to interpret sensors and respond using transducers of your choice.

*Note: *open source means free to inspect & modify*

What are the specifications of the Arduino?

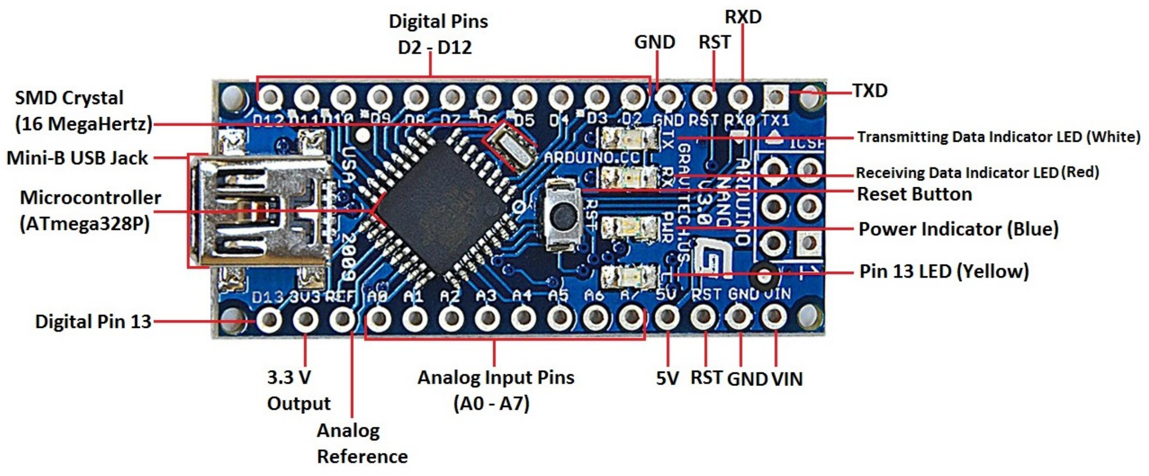
- 8 kBytes of Flash program memory
- 1 kByte of RAM
- 12 MHz (Apple II: 1 MHz)
- Inputs and Outputs: 13 digital input/output pins and 5 analog input pins

Support

In the Libraries available under 'Files' in the IDE there are a large number of programmes already written that are open source and may be used. In addition many online forums are available to guide you with your programming. Some examples of Arduino hub tutorials can be found at:

- <https://www.arduino.cc/en/Tutorial/HomePage >
- Instructables < https://www.instructables.com/class/Arduino-Class/>

What is the layout of your Arduino Nano?

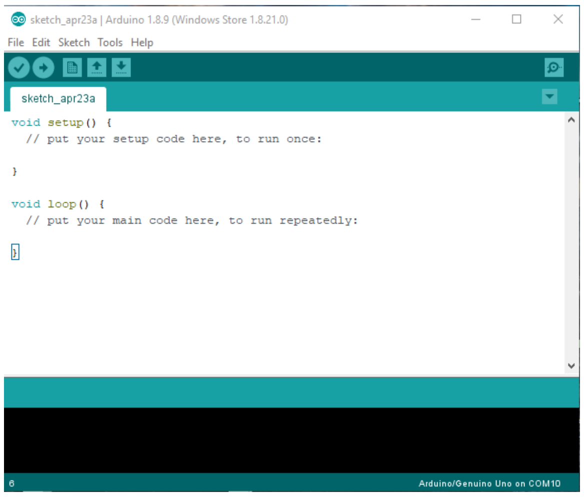


Arduino Nano V3.0 Pinout

Arduino IDE

Download the IDE from the Arduino site before the Innovation Build Day.

This can be done from < https://www.arduino.cc/en/Main/Software>. Choose the version to suit your computer. When you open the IDE the programming environment will look like this:


sketch_apr23a | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
sketch_apr23a

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

6   Arduino/Genuino Uno on COM10

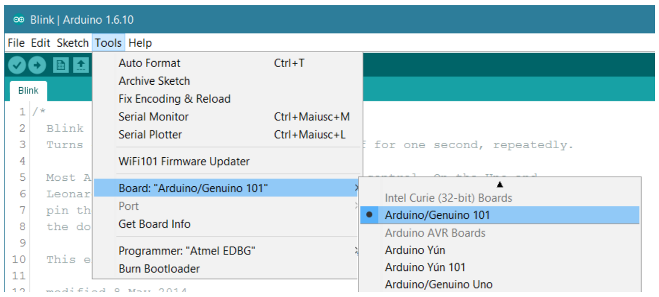
A program (or Sketch) is separated into 2 parts

1) The **void setup** is where you define the pins you will be using etc and any variables

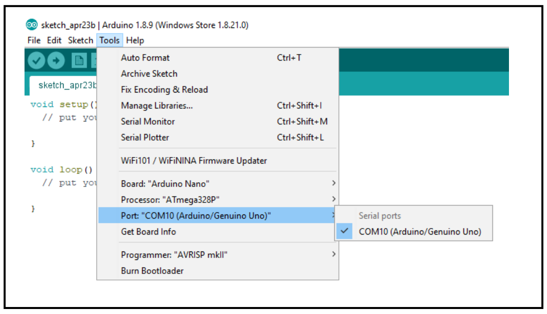
2) The **void loop** is where the active program will be written.

Getting Started

1) Plug in the USB cord into your Arduino board and computer
2) Go to Tools --- Board and select the board you are using – the Arduino Nano

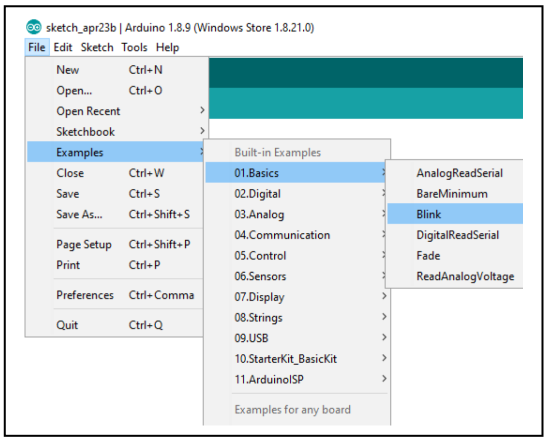


Now check and select the port the USB is connected to by going to Tools----Port
The IDE should automatically select the correct Port. If it doesn't, then go to Ports in 'Windows Device Manager' and select the correct one. You are now set up ready to start programming.

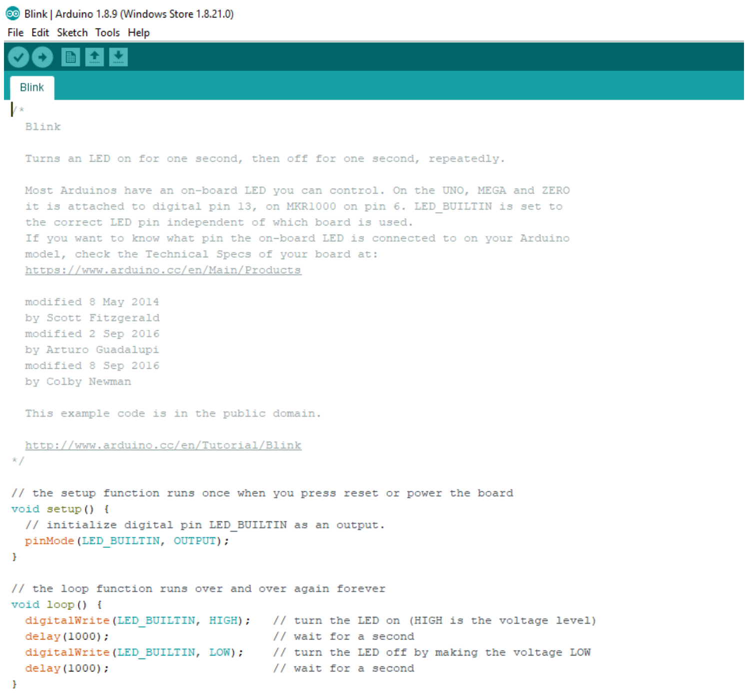

sketch_apr23b | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Nano"
Processor: "ATmega328P"
Port: "COM10 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM10 (Arduino/Genuino Uno)

Let's start with a stored program that has already been written for us to flash an LED.

It is found under File ----Examples ----Basics ---- 'Blink' as shown



Program (usually called a **Sketch** by Arduino programmers)


Blink | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
Blink

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Note the initial blurb is explanatory and doesn't get downloaded because it is sandwiched between the /* and */ symbols

Similarly the **Comments** that explain what a line of code does that immediately follow the // symbols are not downloaded into the Chip. These are very useful to show what you are trying to achieve in each line or part of your program.

Arduino programmers call such a program a **'Sketch'**

Downloading

You could verify firstly your code by clicking on the **Tick** on the tool bar. A report will appear at the bottom of the page. It will appear in orange if there is a fault in what you have written and will need to be corrected before it will work.

Or, you could just click on the **Right Arrow** and this will **download the programme into the chip.** A **Progress bar** will be shown in the green bar at the bottom right of the screen.

Where is the LED??

Youi don't need to connect one as we have used a built in LED on Pin 13. You should now see it flashing.

Program explanations*:*

In void setup

- the Pinmode line names the LED_BUILTIN pin and defines it as an OUTPUT. Output means that current will runout of them to drive devices.

In void loop

- The digitalwrite command defines the state of the pin ie HIGH means the high digital state exists (High voltage on the pin)
- The delay command simply says to hold this for 1000ms or 1s
- The digitalwrite command now sets the pin to the digital low state (zero voltage) so low (0V) voltage on the pin means that no current is driven out to light the LED
- The last delay command defines how long it holds this last state before looping back to the start

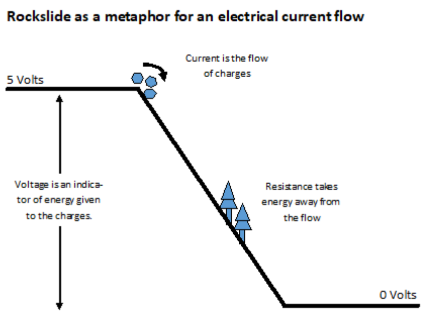
Try changing the delay time and then downloading again.

Circuitry Concepts

What is happening on the Arduino Board?

For an electrical circuit to work properly there must be;

1) A complete circuit.
2) An energy difference between a high voltage point and a lower one – usually provided by an energy source such as a battery.
3) Electrical components that the charges pass through that resist flow and take away some of their energy.

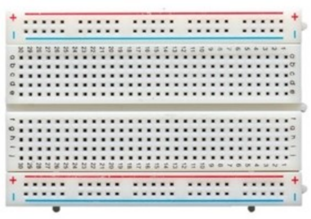

Rockslide as a metaphor for an electrical current flow
5 Volts
Current is the flow of charges
Voltage is an indicator of energy given to the charges
Resistance takes energy away from the flow
0 Volts

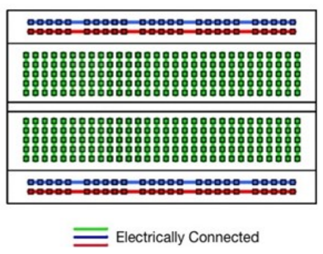
Current will flow from a high voltage point to a lower one in a complete circuit.

Breadboard connections

It is easy to connect components using a breadboard as no soldering is required and you can easily rearrange circuits.

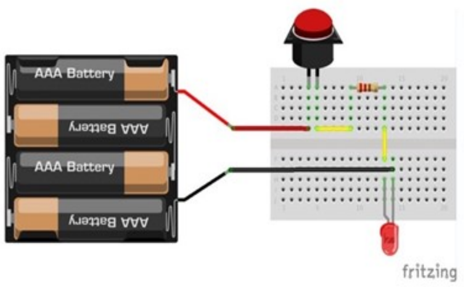
In the second diagram, the coloured lines show how holes are connected underneath the board.

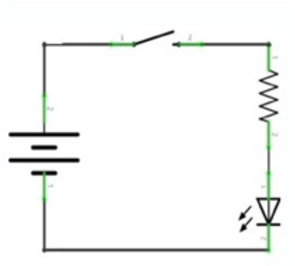




Try connecting the battery pack, switch resistor and LED like this:

The resistor is of size 220 Ohm (Red, Red, Brown). While resistors can be connected either way around (non-polar) the LEDS have a positive side = long leg and a negative leg short leg (because it is minus a bit). The negative side goes closest to the batteries "–" terminal (black lead), and positive leg closest to the "+" side of the battery (red lead).





Note: Red LEDs are damaged if charge with more than 2V pass through them.
If the battery pack puts out 6V, what device in this circuit removes the potentially damaging 4V?

Using these concepts to connect a separate LED for our Blink programme:

Connect the red LED from your kit between Port D2, through a 200 Ohm resistor to Ground. The Output pin D2 will provide a Voltage source of 5V and Ground 0V. The pins are therefore acting like a battery in the circuit.

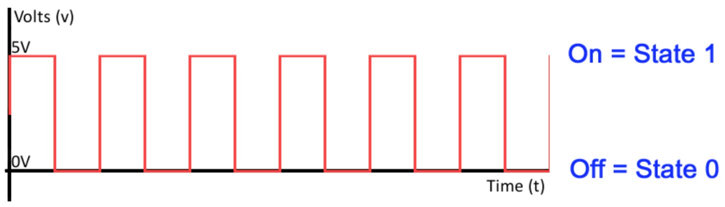
The resistor will remove the potentially damaging voltage.

Adapting the Sketch:

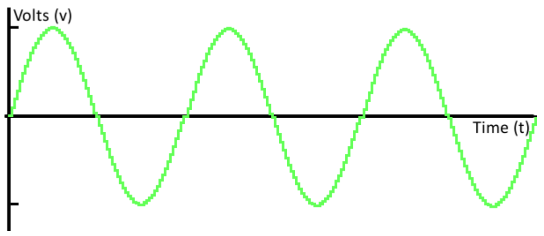
```
void setup() {
   pinMode(11, OUTPUT);    // Set up pin 11 as an output.
}

void loop() {
  digitalWrite(11, HIGH);    // turn the LED on by writing a 'Logic HIGH' (5V to pin 11
  delay(1000);               // wait for 1s
  digitalWrite(11, LOW);     // turn the LED off by writing a 'logic LOW' (0V to pin 11
  delay(1000);               // wait for 1s and repeat
}
```

Analogue and Digital

What is the difference between Digital and Analogue values. The Arduino has a range of pins that are Digital and some that are Analogue as shown in the following 'Pinout' diagram which also shows other important parts of the Arduino board we will be using.



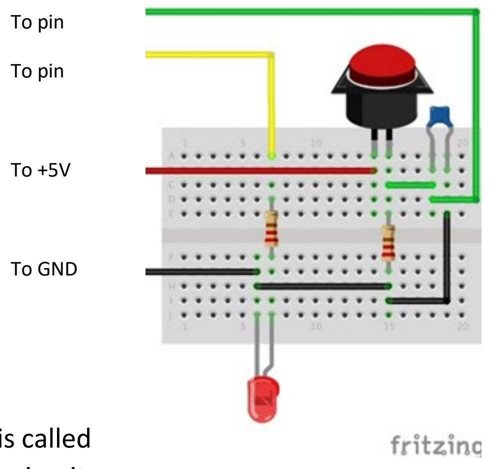
Digital values can only be On or Off. These binary states are 1 and 0.



Analogue values can be any value between the maximum and minimum states.

All pins on the chip are naturally digital but some are designated as Analogue because they can convert analogue values into digital using an ADC (Analogue to Digital Converter). The converter has 10 bit resolution meaning that it can digitise the max value available into 2^{10} = 1024 steps.

Using a Digital Input to turn on the LED



A ***switch is a digital input*** as it has an on/off output. We can set this up so that when pressed it connects to 5V which is logic HIGH and when released it has 0V or logic LOW. Write a program to light up an LED when button is pressed.

Circuit

Connect as shown. The resistor sizes are both 220 Ohm (Red, Red, Brown).

When the switch is closed it connects directly to pin 12 = logic HIGH

The resistor attached to the switch ensures the pin voltage is tied down to GND ordinarily waiting to change to digital HIGH. The resistor is called a 'pull down resistor' to stop the pin voltage floating up and turning the circuit on accidentally. Although connected to ground ordinarily, the pin prefers the zero resistance connection through the switch when it is pressed. This then makes the pin HIGH.

The capacitor stops 'debounce' where the switch press may result in a double connection or bounce. Switches seldom give a clean contact.

Duleepa's Sketch:

```
const int LEDPin = 11;                  //LED is connected to pin 11
const int switchPin = 12;               //Push button switch is connected to pin 12
int switchInput;                        //Declare a variable to hold the input from switch

void setup() {
  pinMode(LEDPin,OUTPUT);               //Initialise the LED pin as an output
  pinMode(switchPin, INPUT);            //Initialise the push button pin as an input
}

void loop(){
  switchInput = digitalRead(switchPin); //Read the input from the switch
  digitalWrite(LEDPin,switchInput);     //Write the value in switchInput to LEDPin and repeat
}
```

What would you change in the circuit & programme to get the switch to turn **on** the LED when it connects to 0V?

Note: This would be an 'active LOW' switch

Decision Making

'if' and *'else'* statements can be used to make decisions.

Logic operators compare variables to enable these decisions

- Equal (A==B),   not equal (A!=B)
- Greater than (A>B),   Less than (A<B)
- AND (A&&B),   OR (A||B)

For our previous circuit we could alter the program to be able to stay on when switch is pressed and go off when pressed again.

Sketch

```
const int LEDPin = 11;                  //LED is connected to pin 11
const int switchPin = 12;               //Push button switch is connected to pin 12
int switchInput;                        //Declare a variable to hold the input from switch
int flagA = 0;                          //Declare a variable  as a flag and initialise to 0
```

```
void setup() {
  pinMode(LEDPin,OUTPUT);                    //Initialise the LED pin as an output
  pinMode(switchPin, INPUT);                 //Initialise the push button pin as an input
}

void loop(){
  switchInput = digitalRead(switchPin);      //Read the input from the switch
  if (switchInput==1 && flagA==0){           //If push button pressed and flagA is 0
  digitalWrite(LEDPin, !digitalRead(LEDPin)); //Toggle LEDPin
  flagA = 1;                                 //set the flag to 1 to avoid toggling if button held on
  }
  if (switchInput==0) {                      //If push button is released
    flagA = 0;                                //Reset the flag to 0 and wait for a button press
  }
}
```

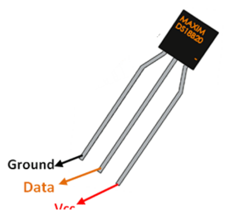
Temperature measurement?

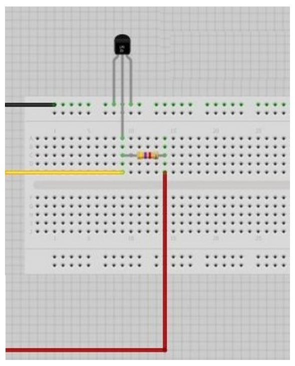
For this we will use the Dallas one wire temperature sensor DS18B20

Circuit connection:

Note: The resistor value is 4.7k Ohm.

The connections of the DS18B20 are as shown. The flat side is face up





Sketch:

by m.vasilakis Tutorial at the address below

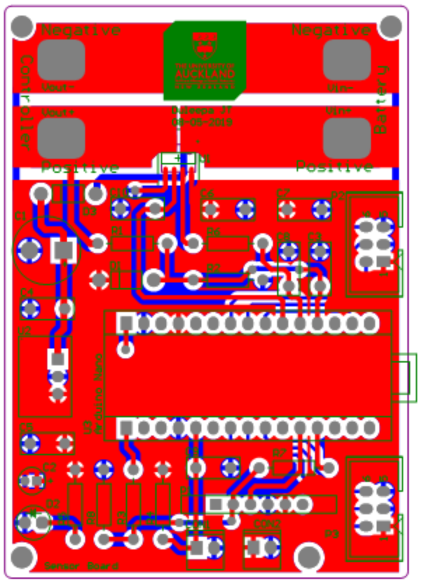
```
/* Arduino DS18B20 temp sensor tutorial
   More info: http://www.ardumotive.com/how-to-use-the-ds18b20-temperature-sensor-en.html
   Date: 19/6/2015 // www.ardumotive.com

_____________________________________________________________________________*/

#include <OneWire.h>                        //Include libraries
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 2                      // Data wire is plugged into pin 2 on the Arduino
OneWire oneWire(ONE_WIRE_BUS);              // Setup a oneWire instance to communicate with Dallas temperature IC
DallasTemperature sensors(&oneWire);        // Pass our oneWire reference to Dallas Temperature.


void setup(void)  {
  Serial.begin(9600);                       //Begin serial communication
  Serial.println("Arduino Digital Temperature  // Serial Monitor Version"); //Print a message
  sensors.begin();
}

void loop(void)  {
  sensors.requestTemperatures();            // Send the command to get temperatures
  Serial.print("Temperature is: ");
  Serial.println(sensors.getTempCByIndex(0)); // Why "byIndex"? You can have more than one IC on the same bus. 0 refers to the first IC on the wire
  delay(1000);                              //Update value every 1 sec
}
```

Using the University of Auckland Mounting Board

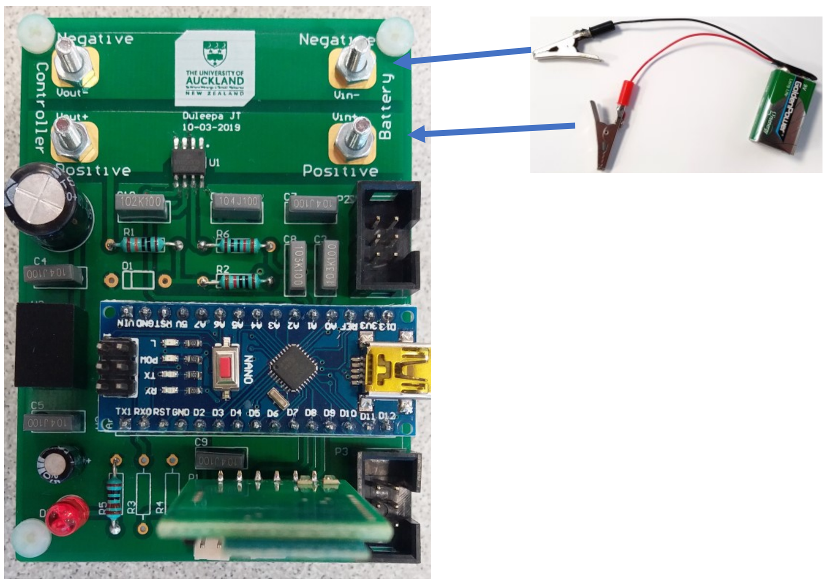
We are grateful to the University of Auckland for designing this board that hosts the Arduino Nano, Bluetooth module and has facility to operate with a 9V battery or with the 24V of the vehicle. During our workshop we can use the board powered by a 9V battery connected as shown from now on.

Insert the Nano into the UoA board as shown
Attach the 9v battery as shown
Insert the Bluetooth module into header labelled P1

The circuitry labeled "5V Regulator", converts the 18-26 V battery voltage down to 5 V in order to power the Arduino Nano, the Bluetooth module and the current sensor IC as supplying these directly from your 24V battery will damage them.



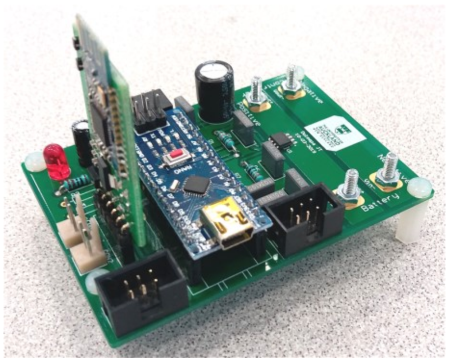
PCB layout



Loaded with components *9V Battery connection*

Side on the board looks like this:

Note the vertically mounted Bluetooth module



Current & Voltage measurement

As the brain of the system the Arduino Nano can read and interpret battery voltage, current and could use these to calculate drive power, or remaining battery power, battery temperature etc and then send it out to your phone via the Bluetooth module.

Voltage

The Analogue pins (A0 to A7) of the Arduino Nano, allow it to read an Analogue voltage that is between 0 V and 5 V and convert to a digital number that is between 0 and 1023 (1024 steps)

So if you apply 5V the digital number would be 1023
2.5V the digital number would be 512
0V the digital number would be 0
Etc

We can convert these digital numbers into the correct values for whatever measurement we are making
For example, if you apply 2.5 V to A0, it will be converted to the digital number 512 (i.e. 1024 x 2.5/5).

We can use two Analogue pins to read the battery voltage and battery current and convert to digital numbers.

If you are using a 24V battery supplying this directly to an Analogue pin of the Arduino Nano will damage it so, the two resistors labeled “R1” and “R2” are used to step the battery voltage to below 5 V.

Eg. , if “R1” is 9 k and “R2” is 1k, then 24V will be stepped down to 2.4V (= 24x1/(9+1))

The capacitor labeled “C3” will help filter out any ‘noise’ (erratic variation) there may be in the voltage stepped down by “R1” and “R2” before it is measured by pin A0 of the Arduino Nano.

Current

As the Arduino Nano can only read Analogue the current needs to be converted to a voltage. To do this we use the integrated circuit (IC) labeled “U1 ACS730”. It will measure the current flowing from the battery to the motor controller and convert this current to a voltage between 0 V and 5 V.

How does it do this?
The current sensor IC produces 2.5 V when the current is 0 A, and if the current is positive 0.04 V is produced per 1 A flowing through it. For example, if the current flowing through it is 20 A then the voltage produced by the sensor will be 3.3 V (the 3.3V reading = 2.5 V + 20 x 0.04 V)
So Current = 2.5 +

The resistor “R6” and the capacitor “C8” are used to filter any ‘noise’ there may be in the voltage produced by “U1” before it is measured by pin A1 of the Arduino Nano. The current sensor IC may not produce exactly 2.5 V when current is 0 A. This value is made available to measure on pin A2 of the Arduino Nano so we can further improve the accuracy of the measurement.

Shown below is Duleepa’s program to read the voltages at pins A0 and A1 and then performs calculations to determine the actual battery voltage and current.

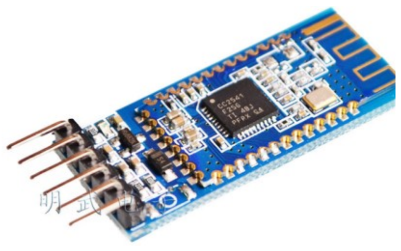
Sketch:

```
void setup() {
  Serial.begin(9600);                                   // Setup serial communication with a PC via USB connection
  Serial.println("Serial Coms with PC Initiated");
  Serial.println("");
}

void loop() {
  float Voltage = (10*5)*analogRead(A0)/1024;           // Read voltage on A0 and convert to actual voltage
  float ZeroCurrentV = 5*analogRead(A2)/1024;           // Read the current sensor output at 0 A as it may
                                                        // be slightly different to 2.5 V
  float Current = 25*((5)*analogRead(A1)/1024 - ZeroCurrentV);  // Read current on A1 and convert to actual current
  Serial.print("Voltage: ");                            // Display both voltage and current on the PC
  Serial.println(Voltage);
  Serial.print("Current: ");
  Serial.println(Current);
}
```

Suggestions for other sensor connections (Temperature, Tachometer, Low battery warning, etc)

- **Tachometer:** The connector pins labeled “CON1” and “CON2” can be used to connect a tachometer to this board.
- **Temperature:** Header pins labeled “P2” can be used to connect other Analogue sensors like a temperature sensor.
- **Low Battery indicator etc:** Header pins labeled “P2” can be used for digital inputs/outputs like a battery low indicator.

Bluetooth Communication



The Bluetooth unit that comes with this kit is a HM-10 variant that handles both Rx and TX. It communicates with the Arduino Nano using serial. Unfortunately we have used the built-in hardware serial communication port of the Arduino Nano to communicate with the PC through the USB cable we are using. This means we need to find another serial communication port to connect the Bluetooth module. We can use a software serial communication port (basically using two standard digital pins of the Arduino Nano) to emulate a serial port. The software we will use to do this is called **AltSoftSerial**

To do this go to Tools > Manage Libraries, find AltSoftSerial and install it so we can use it in our project. As shown by the counter example below, setting up AltSoftSerial and using it to send messages is similar to what you have done before with Serial.Begin() and Serial.Print()

The Bluetooth module in the kit needs to be connected to header pins labeled “P1” which provides 5 V and GND to the Bluetooth module and also connects the transmit pin of the AltSoftSerial port on the Arduino Nano (digital pin “D9”) to the Bluetooth module.

The resistors “R7” (3.3 k) and “R8” (5.6k) step-down the 5V serial transmit signal generated by the Arduino Nano to 3.3V to avoid damaging the Bluetooth module.

The receive pin of the AltSoftSerial port on the Arduino Nano, which is digital pin “D8” is also connect to the Bluetooth module in case you like to send messages to the Arduino Nano.

Note:

I. A Bluetooth Serial software application can be download to view the message sent by the Arduino on the phone.
II. Once you pair the Bluetooth module with the phone via the Bluetooth Serial app, the blinking red light on the module will stay on indicating successful pairing

Sketch:

```
#include <AltSoftSerial.h>
AltSoftSerial BTserial;
int counter = 0;                                  // Setup a variable to hold a count value for our counter

void setup() {
 Serial.begin(9600);                              // Setup serial communication with a PC via USB connection
 Serial.println("Serial Coms with PC Initiated");
 Serial.println("");
 BTserial.begin(9600);                            // Setup serial communication with BLE using AltSoftSerial
 Serial.println("Serial Coms with BLE Initiated");
}

void loop() {
 Serial.print("Counter Value: ");                 // Display the value of counter on PC
 Serial.println(counter);
 BTserial.write(counter);                         // Transmit the value of counter via BLE to phone
 counter = counter+1;                             // Increment the counter by 1
 if (counter>10)                                  // If counter has reached 10 then reset it to 0
 counter=0;
 delay(1000);                                     // Wait for 1000 ms before repeating above steps
}
```

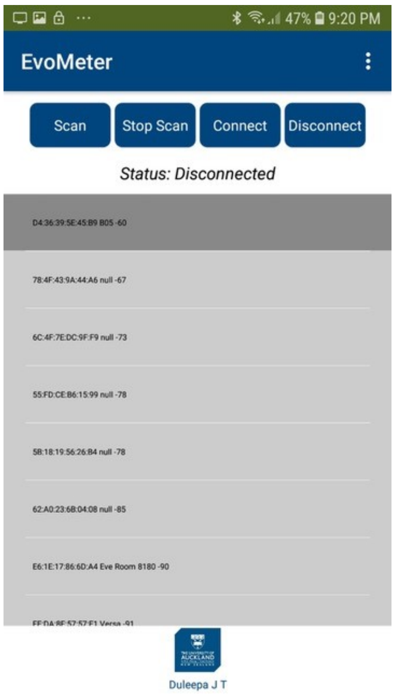
Developing a Mobile Application

To display the measurements on your phone we need an App. The App Inventor tool (http://appinventor.mit.edu/explore/) developed by MIT in USA is one of the easiest options to use. Get started by creating an account and importing the file provided to get started. This file already implements part of the App needed. Now Run this App on the phone by downloading the App to the phone.

Press the scan button to scan for the Bluetooth module. Once found, select stop scan and connect to pair the phone with the Bluetooth module. Now observe the voltage reading getting incremented from 0 to 10

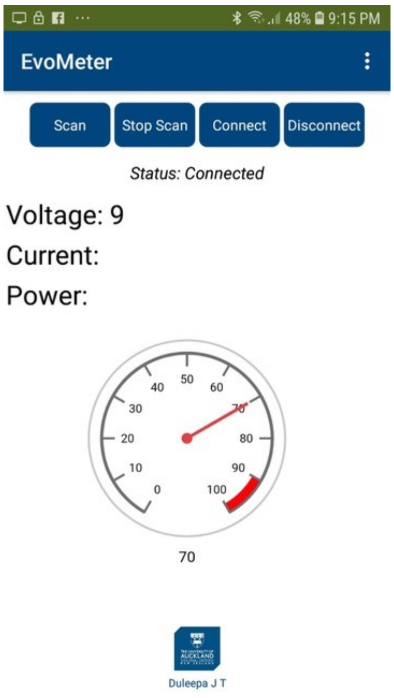
Note that we are using a Internet of Things (IOT) platform called ThingSpeak (https://thingspeak.com) to also upload the data to the web and display this remotely. ThingSpeak is free and an account can be created with them to setup your own web based data logging and display.

Try modifying the App to meet your requirements :o))



Status: Disconnected

D4:36:39:5E:45:B9 B05 -60
78:4F:43:9A:44:A6 null -67
6C:4F:7E:DC:9F:F9 null -73
55:FD:CE:B6:15:99 null -78
5B:18:19:56:26:B4 null -78
62:A0:23:6B:04:08 null -85
E6:1E:17:86:6D:A4 Eve Room 8180 -90
FF:DA:8F:57:57:F1 Versa -91



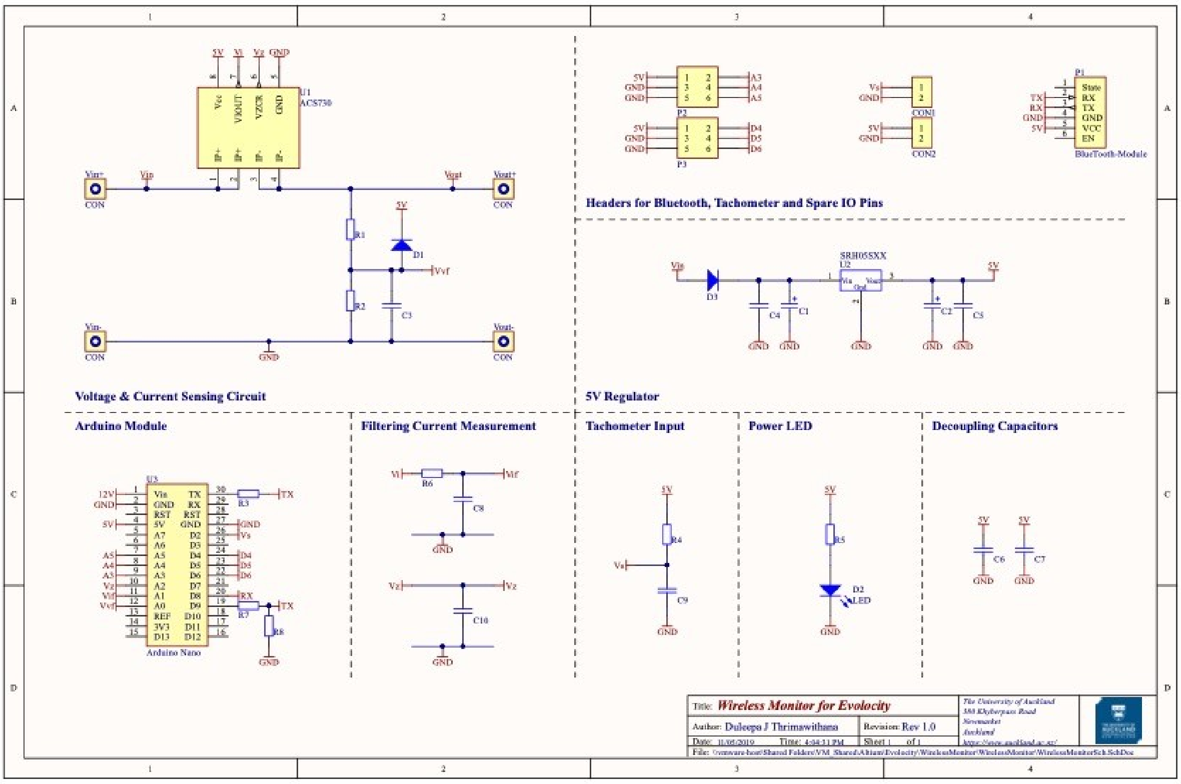
Status: Connected

Voltage: 9

Current:

Power:

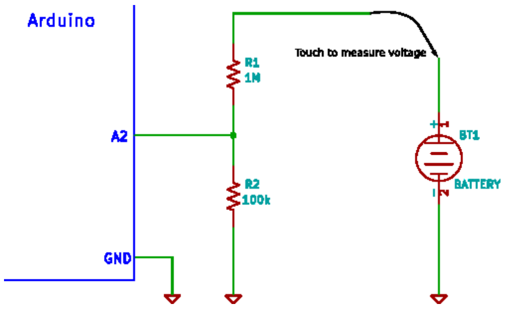
Header connections etc



Reading Voltages *(OPTIONAL approach without UoA board)*

Circuit for sensor

The following circuit uses low cost resistors to make a Voltage Divider that could measure battery voltage up to 30V. The impedance of this system offers short circuit protection to the Arduino.



The voltage of this battery is being tested.

Note the touch probe and common ground

Connect up this sensor circuit from the parts in your kit.

Sketch **(by WA Smith from address below):**

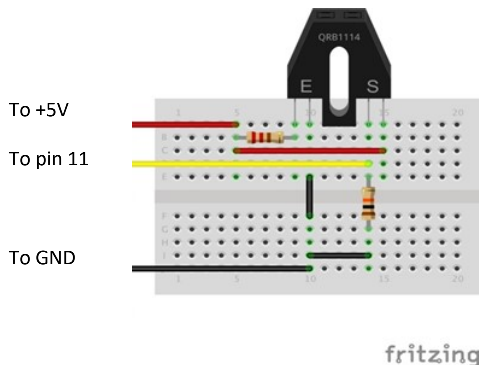
```
/*---------------------------------------------------------------
Description:    Reads value on analog input A2 and calculates the voltage assuming that a voltage divider network on the pin divides by 11.
Hardware:       Arduino Uno with voltage divider on A2.
Date:           22 May 2013
  Author:       W.A. Smith, http://startingelectronics.org
--------------------------------------------------------------*/
#define NUM_SAMPLES 10          // number of analog samples to take per reading
int sum = 0;                    // sum of samples taken
unsigned char sample_count = 0; // current sample number
float voltage = 0.0;            // calculated voltage

void setup()  {
   Serial.begin(9600);
}

void loop()  {
      while (sample_count < NUM_SAMPLES) {
         sum += analogRead(A2);   // take a number of analog samples and add them up
         sample_count++;
         delay(10);
   }
                                // calculate the voltage
                                // use 5.0 for a 5.0V ADC reference voltage
                                // 5.015V is the calibrated reference voltage

   voltage = ((float)sum / (float)NUM_SAMPLES * 5.015) / 1024.0;
                                // send voltage for display on Serial Monitor
                                // voltage multiplied by 11 when using voltage divider that
                                // divides by 11. 11.132 is the calibrated voltage divide
                                // value
   Serial.print(voltage * 11.132);
   Serial.println (" V");
   sample_count = 0;
   sum = 0;
}
```

This adds 10 samples together and stores them in ***sum*** *Variables are reset after displaying the voltage*

The sum of 10 samples is divided by 10 and then multiplied by the reference voltage on pin A2. This value should be measured and the correct value entered in this line.

Calibration:
Similarly measure the size of each resistor. The ratio of total of the series resistance divided by the smaller resistors value will be close to 11.000 but the exact value

Optical Tachometer Solution (Optional)



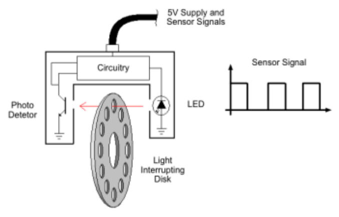
Using the photo Interrupter in your kit to measure rotational speed.

Circuit:
Connect to the Arduino as shown.

The Infra Red LED is supplied with 5V through a current limiting 220 Ohm resistor (Red, Red, Brown).

The resistor between pin 11 and ground is a 10k Ohm (Brown, Black, Orange)

The program will set up pin 11 as an input. It will then wait for logic high and measure the duration of the logic high signal and this will be related to disc or wheel speed.



Duleepa's tachometer Sketch:

```
const int tachoPin = 11;                //Tachometer is connected to pin 11
const int wheelRadius = 150;            //Radius of wheel in mm

void setup()  {
  pinMode(tachoPin, INPUT);             //Initialise the Tachometer pin as an input
}
void loop() {
  long timeHigh = 0;                    //Variable to store time tacho is high for in microseconds
  long carSpeed = 0;                    //Variable to store cart speed in kph
  timeHigh = pulseIn(tachoPin, HIGH);   //Measure length of tacho high pulse (active low)
  if (timeHigh==0) {                    //If no tacho pulse the speed is 0 kph
  carSpeed = 0;
  }
  else {                                //Else speed is 2*pi*wheelRadius*3600/(timeHigh*correction)
    carSpeed = 20000*wheelRadius/timeHigh;
  }
}
```