Working with Arduino

Workbook
Welcome to Working with Arduinos @UQ Workshop

In this workbook, you will find information on how to use an Arduino. You can keep this workbook and bring it home. The information may be useful if you want to continue working with Arduinos when you are back at your school.

REMEMBER to always unplug the USB when working on setting up the arduino. Check with tutors before plugging in the USB.
Working with Arduinos@UQ is an initiative of the School of Information Technology and Electrical Engineering (ITEE) at The University of Queensland (UQ), which aims to introduce high school students to working with Arduinos and to the application of Arduinos.

Working with Arduinos@UQ is designed for secondary school students, and consists of a 3 hour workshop that allows the students to get a hands-on experience of wiring and programming an arduino.
# USING AN ARDUINO

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADVANTAGES TO USING ARDUINO’S</td>
<td>5</td>
</tr>
<tr>
<td>BREADBOARD</td>
<td>6</td>
</tr>
<tr>
<td>THE WIRING APPLICATION</td>
<td>6</td>
</tr>
<tr>
<td>INPUTS AND OUTPUTS</td>
<td>7</td>
</tr>
<tr>
<td>SENSORS</td>
<td>7</td>
</tr>
</tbody>
</table>

## TUTORIALS

<table>
<thead>
<tr>
<th>Tutorial</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLINK (😊)</td>
<td>8</td>
</tr>
<tr>
<td>RGB LED (😊😊)</td>
<td>11</td>
</tr>
<tr>
<td>FADING RGB LED (😊😊😊)</td>
<td>13</td>
</tr>
<tr>
<td>TONE (😊)</td>
<td>16</td>
</tr>
<tr>
<td>TONE ON BUTTON PRESS (😊😊😊)</td>
<td>18</td>
</tr>
<tr>
<td>TONE WITH FLEX SENSOR (😊😊😊😊)</td>
<td>20</td>
</tr>
</tbody>
</table>

## RESOURCES

22
USING AN ARDUINO

An Arduino is a prototyping tool for sensing and controlling the physical world. It is an open-source physical platform, based on a simple micro controller board and a development environment for writing software on the boards. An Arduino is breakout board for a microcontroller, which is a very small computer.

An Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors or other physical outputs. Arduino projects can be stand-alone, or they can communicate with software running on your computer (eg. Flash, Processing, MaxMSP). The boards can be assembled by hand or purchased pre-assembled; the open-source IDE can be downloaded for free.

The programming environment used for programming an Arduino is based on the language used for Wiring (a similar physical computing platform).

Advantages to using Arduino’s

- Can be run independently of a computer
- Can control PWM, voltages etc. More functionalities than Phidgets
- Can communicate with many programming languages
- Useful when hacking into commercial appliances
Breadboard

We are using a solder-less breadboard in the workshop for creating circuits. Solder less breadboard means the breadboard can be re-used. When you connect components in your breadboard you need to make sure you connect the pins of your components to separate conductive rails (these are shown below).

The Wiring Application

Wiring is the programming application that you will use to program the Arduino. The programming language is primarily based on Processing, which is a derivative of Java. The process for making an Arduino application is:

• Write the code
• Compile the code
• Upload the code to the Arduino board

Sometimes you want to interact with a specific piece of hardware. If you can find a library for the hardware, it will be easier to interact. [http://www.arduino.cc/en/Reference/Libraries](http://www.arduino.cc/en/Reference/Libraries)


In summary, Wiring has two components:

• setup(), and
• loop()

The setup() function is run when the application is first started. Any code or variables you want to set up and only run once should be set in here.

The loop() function is an infinite loop which runs the code inside, as fast as it can. Because of this, it is usually necessary to put in a delay(‘1000’), to ensure that the board does not crash.
Wiring does not include too much functionality, as it is important to keep the complexity of the code to a minimum. Wiring does not support classes due to this. Read the reference at: http://www.arduino.cc/en/Reference/Extended to ensure that what you are trying to do can be done.

If you require doing more complex operations, or communicating with a computer, you need to use the Serial Communication functionality. There are tutorials online for talking through the Serial Communication with most programming languages. We recommend using Flash, as it is fairly simple to integrate (compared to other languages). An Actionscript 3 library for connecting Arduino’s can be found here: http://code.google.com/p/as3glue/

**Inputs and Outputs**

The Arduino uses inputs to determine what is happening in the environment, and outputs to respond to the inputs. Inputs are any kind of sensor and outputs can be LEDs, speakers, motors, etc.

**Sensors**

Sensors are a way to detect something. There are hundreds of possible sensors, some can be built, a lot we have, some may have to be bought. Sensors can either input an analogue (a) signal (range 0-255) or a digital (d) signal, which has an input of either on or off, 0-1.

Some sensors:
- temperature (a)
- accelerometer (think iphone/wiimote) (a)
- touch sensor (a / d)
- pressure sensors (a)
- tilt sensors (a)
- push sensors (a)
- proximity sensors (a)
- light detection sensors (a / d)
- infrared sensors
- rotation sensors (one turn)
- rotation sensors (multiple turns)
- vibration sensors (a)
- humidity sensors (a)
- magnetic sensors (a)
- sonar sensor (a)
- iPod touch wheel (a)
- biometric sensors (a), heart, muscle, eye, skin, brain
TUTORIALS
This booklet contains a few good tutorials, ranging from easy (😊) to hard (😊😊😊😊😊).

Blink (😊)
In this tutorial you will make an LED blink on the Arduino, and then wire up an external LED and make that blink. This tutorial can be found at: http://arduino.cc/en/Tutorial/Blink.

In this tutorial you will need:
- Arduino and programming cable
- Breadboard and wires
- Any LED
- 330Ω resistor

In the Arduino application open the Blink sketch (File -> Examples -> Basics -> Blink). You should now see the code for this tutorial, shown below.
The `setup()` function is run at the start of the program and uses the `pinMode()` function to set pin 13 as an **output**. This is so we can power our LED.

The `loop()` function runs the code inside of it infinitely. In this tutorial we are using the `digitalWrite()` and `delay()` functions. The `digitalWrite()` function writes a digital value (HIGH or LOW) to the specified pin (in this case, pin 13). HIGH will turn your LED on and LOW will turn your LED off. The `delay()` function pauses the code for the specified milliseconds (1000 = 1 second).

The Arduino already has an LED connected to pin 13 on the board and so running the code should make this LED blink.

*Run the code by pressing the Upload button (circled below). This will compile and upload the code to your Arduino board.*

See a tutor if you don’t see a flashing LED or if you have errors.

Now we are going to make our own LED circuit. To do this, you need your breadboard, LED, resistor and wires.

**Remember to UNPLUG your Arduino from your computer BEFORE connecting it to your circuit.**

LEDs are polarized and so it is important which way they are connected in your circuit. If you look inside the LED you can see an anvil shape (the electrode, which is negative) and a smaller bit of metal (the anode, which is positive). You will notice that the LED has a longer leg, which is generally the anode.

Put your LED anywhere in your breadboard but keep track of where the longer leg is. Wire one end of your resistor to the **short** leg of your LED, and the other leg to **GND**, using your breadboard and the wires. Add another wire from **pin 13** to the **long** leg of your LED. The schematic is shown below.

See a tutor if you are not sure how to wire it all together.
Once you are sure that you have wired it correctly, connect your Arduino to your computer.

See a tutor if your LED does not flash.

Exercises:

1. Make your LED flash faster or slower.
2. Make your LED flash in a pattern.
RGB LED (😊😊)

In this tutorial you will wire up an RGB LED and make different colours.

In this tutorial you will need:
- Arduino and programming cable
- Breadboard and wires
- RGB LED
- 3x 330Ω resistors

An RGB LED acts as if it were three LEDs combined into one. You will notice that the RGB LED has 4 pins instead of 2. Three pins are the red, green and blue pins and the fourth pin connects to ground. Any time you use an LED you need a resistor between the LED and the voltage source, and so for an RGB LED you will need 3.

Wire the circuit as shown below.

Remember to UNPLUG your Arduino from your computer BEFORE connecting it to your circuit.
Now we will program our RGB LED. In the Arduino application, type in the software as shown below.

```c
int red = 11;
int green = 12;
int blue = 13;

void setup()
{
pinMode(red, OUTPUT);
pinMode(green, OUTPUT);
pinMode(blue, OUTPUT);
}

void loop()
{
digitalWrite(red, HIGH);
digitalWrite(green, HIGH);
digitalWrite(blue, HIGH);
}
```

In the `setup()` function we set the mode of your RGB LED pins as outputs using the `pinMode()` function. This is because we want to give each pin a voltage to turn them on.

In the `loop()` function we turn our RGB LED pins all to HIGH. This will turn the red, green and blue legs on.

*Upload the program to your Arduino by pressing the Upload button.*

*See the tutor if your LED does not turn on or you have errors.*

**Exercises:**

1. Make your LED flash every second.
2. Make your LED flash RED, GREEN, BLUE.
3. Make your LED flash different colours.
Fading RGB LED (😊😊😊)

In this tutorial you will wire up an RGB LED and make it fade between colours.

In this tutorial you will need:
- Arduino and programming cable
- Breadboard and wires
- RGB LED
- 3x 330 Ω resistors

Wire up the RGB LED as described in the RGB LED tutorial in this workbook. To make the LED fade in and out we need to wire the red, green and blue legs to the PWM pins on the Arduino. This is so that we can output a range of values to our LED. In the previous tutorial we were only outputting either HIGH (to turn the LED on) or LOW (to turn the LED off). The final circuit is shown below.

To make the LED fade we need to give it an incrementing and decrementing set of values. We do this in software by changing the brightness level that we give to the LED. This is changed in the set_brightness() function that we will create in our program. The direction of fade is set using the set_fade() function that we will also create in our program.

The full program is shown on the next page.
sketch_sep25a §

// define the pins for the R, G and B legs of the RGB LED
int red = 9;
int green = 10;
int blue = 11;

// define the initial brightness level of each leg
int brightnessRed = 0;
int brightnessGreen = 0;
int brightnessBlue = 0;

// define the initial fade amount of the LEDs
int fadeRed = 1;
int fadeGreen = 3;
int fadeBlue = 5;

void setup(){
  pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(blue, OUTPUT);
}

// function to set the brightness level of a given pin
int set_brightness(int brightness, int fade){
  return brightness + fade;
}

// function to reverse the direction of the fade amount
int set_fade(int brightness, int fade){
  if (brightness <= 0 || brightness >= 255){
    fade = -fade;
  }
  return fade;
}

void loop(){
  // output the brightness level to each leg
  analogWrite(red, brightnessRed);
  analogWrite(green, brightnessGreen);
  analogWrite(blue, brightnessBlue);

  // change the brightness level for the next time through
  brightnessRed = set_brightness(brightnessRed, fadeRed);
  brightnessGreen = set_brightness(brightnessGreen, fadeGreen);
  brightnessBlue = set_brightness(brightnessBlue, fadeBlue);

  // check if the fade value needs to be reversed
  fadeRed = set_fade(brightnessRed, fadeRed);
  fadeGreen = set_fade(brightnessGreen, fadeGreen);
  fadeBlue = set_fade(brightnessBlue, fadeBlue);

  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
In the `setup()` function we set the mode of the R, G and B pins to be outputs using `pinMode()`.

In the `loop()` function we write the brightness value to each colour pin, update the brightness and reverse the fade direction (if needed).

Upload the program to your Arduino by pressing the Upload button.

See the tutor if your LED does not fade or you have errors.

Exercises:
1. Make your LED fade faster or slower.
2. Make your LED fade between different colours.
**Tone (©)**

In this tutorial you will wire up a piezo speaker and program the Arduino to make a tone.

In this tutorial you will need:
- Arduino and programming cable
- Breadboard and wires
- Piezo speaker

A piezo speaker emits a tone by receiving a pulsed voltage at a desired frequency. The piezo can be wired directly to the Arduino. Wire the circuit as shown below.

To make the piezo speaker emit a tone we need to tell it the frequency of the tone. We will be using the inbuilt `tone()` function to do this. In the Arduino application create a new file and type in the following program.
In the `setup()` function we set the mode of the speaker pin to be an output using `pinMode()`.

In the `loop()` function we make the speaker output a tone by using the `tone()` function. The `tone()` function outputs a 100Hz tone to pin 11.

Upload the program to your Arduino by pressing the Upload button.

See the tutor if your speaker does not make a sound or you have errors.

Exercises:
1. Make a different tone.
2. The `notone(output_pin)` function stops the tone from playing. Try and make your speaker beep on and off.
**Tone on Button Press (😊😊😊)**

In this tutorial you will make a tone play when you press a button.

In this tutorial you will need:
- Arduino and programming cable
- Breadboard and wires
- Piezo speaker
- Push button
- 330Ω resistor

The push button is used to give a HIGH or LOW signal to the Arduino. You can use this to tell the speaker when to play. Our input pins can have noise, which will affect the logic of our code. To stop this we use what we call a *pull-down resistor*, which is the 330Ω resistor connected between the wire that goes to your input pin and ground.

Wire the circuit as shown below.

![Circuit Diagram](image)

See the tutor if you are unsure of the wiring.

We want our speaker to make a noise only when the pushbutton is pressed. We need to be able to tell *when* our button is pressed. We do this by using the `digitalRead()` function. This reads the value on the input pin connected to our push button. If the value is HIGH then we know that the button has been pushed. When our button is pushed, we will make our speaker output a noise by using the `tone()` function. When it has not been pushed, we will turn our speaker off using the `noTone()` function. This is shown in the program below.
In the `setup()` function we use the `pinMode()` function to set the mode of our speaker (pin 11) to be an output and the mode of the push button (pin 12) to be an input. The speaker is an output because we are giving it a signal and the push button is an input because we are reading the value from the button (ON/OFF).

In the `loop()` function we use the `digitalRead()` function to read the value of the pushbutton; whether it has been pushed (HIGH) or not (LOW). We check whether it has by using an `if statement`. If the button has been pressed (reads HIGH) we output a tone to the speaker using the `tone()` function. If the button has not been pressed we turn the speaker off by using the `noTone()` function.

Upload the program to your Arduino by pressing the Upload button.

See the tutor if your speaker does not make a sound when you press the button or you have errors.

Exercises:
1. Make a different tone.
2. Wire up an LED to turn on when you make a tone (hint: when the button is pressed).
Tone with Flex Sensor 😊😊😊

In this tutorial you will be changing the tone of a piezo speaker with a flex sensor.

In this tutorial you will need:
- Arduino and programming cable
- Breadboard and wires
- Piezo speaker
- Flex sensor
- 10kΩ resistor

The flex sensor detects bend or flex. The resistance of the flex sensor changes as it is bent. In our circuit, this changes the input voltage and so we can use this to change the tone of our speaker.

Wire the circuit shown below.

We want to change the tone of our speaker when we bend the flex sensor. We do this by reading in the voltage across the flex sensor, and *mapping* this to a frequency range. The frequency value is then output to the speaker. This is shown in the code below.
The `setup()` function sets the speaker pin to be an output.

The `loop()` function reads the value from the flex sensor using `analogRead()`, converts it to a frequency value between 100Hz and 2000Hz using `map()`, and outputs this to the speaker using `tone()`.

*Upload the program to your Arduino by pressing the Upload button.*

*See the tutor if your speaker does not make a sound when you bend the flex sensor or you have errors.*

**Exercises:**

1. Wire up an LED to change brightness with the flex sensor.
RESOURCES

Arduino Homepage::
http://www.arduino.cc/

Arduino Troubleshooting:

Arduino Playground:
http://www.arduino.cc/playground/

Good series of tutorials:
http://www.ladyada.net/learn/arduino/lesson3.html

How Breadboards work:
http://eecs.vanderbilt.edu/courses/ee213/Breadboard.htm

Beginners Electronics:

Circuit Diagrams:
http://wiring.org.co/learning/tutorials/diagrams.html

Resistance Calculator:
http://www.dannyg.com/examples/res2/resistor.htm

Jody Culkin’s introduction to Arduino
Marnie Lamprecht

Marnie Lamprecht is a PhD scholar in the School of Information Technology and Electrical Engineering at the University of Queensland in the field of biomedical engineering. In Marnie’s scarce spare time she enjoys being outdoors, and although Marnie is not super experienced (yet), she loves renovating.

Andrew Dekker

Andrew (@simultech) is a researcher, designer and developer within the field of interaction design. Andrew is always keeping himself busy, if not by studies, he designs websites and mobile apps. Andrew has all the answers to all your digital problems😊

Marie Boden

Marie Boden is a PhD scholar in the School of Information Technology and Electrical Engineering at The University of Queensland. Marie loves playing with Lego robots and going to the beach.
What is IT/ICT?
Information & Communications Technology is a term that covers all forms of computer and communications equipment and programming software used to create, store, transmit, interpret, and manipulate information in its various formats (e.g., business data, voice conversations, still images, motion pictures and multimedia presentations).

Why ICT?
ICT is an essential part of the way the world operates, so a career in ICT gives you an opportunity to help shape our future and be employed in many different ways and places.

The successful ICT student has good analytical and thinking skills. Some mathematical ability is required, but lateral thinking and a creative mind are widely considered by industry leaders to be essential. By combining technical strengths with creative and innovative ability and sensitivity to the needs of the user, our graduates help to align technology with the needs of business and the community and to deliver rewarding solutions on time and within budget.

Where are the jobs in ICT?
After a slump in job opportunities a few years ago, industry in now concerned that there will be a shortage of qualified people in the next few years. Business leaders predict that the new growth in the ICT industry is more-sustainable and long-term than the boom that preceded the ‘dotcom’ crash of 2000. A 2005 Australian Government report says that IT is one of the top five occupations in terms of employment growth over the next five years. The Department of Employment report also says that IT is now in the second-highest earnings bracket.

The Queensland Government expects Information & Communications employment in the state to grow by 47% this decade. The Government held a national summit in Brisbane in June 2006 to focus on the growing ICT skills shortage. The latest Queensland ICT Industry Survey (2003-4) shows the local industry has strong sales and high full-time employment levels. In July 2006, the Australian newspaper carried a report that ICT-based projects are booming, with Queensland leading the nation. In October 2006, the Melbourne Age newspaper reported that job advertisement growth, as measured by recruitment firm Olivier's internet jobs index, was strongest in the ICT sector, had been for a year, and was expected to continue. And in February 2007, the Australian newspaper reported that ICT job growth (strongest in Queensland) is such that shortages of candidates could soon arise. In May 2007, the Brisbane Times on-line newspaper reported that Google Australia was hiring as many quality software engineers as it could find.

ICT permeates practically every industry, with every medium to large organisation having its own ICT personnel or calling on other businesses to provide the services they need. The variety of work is increasing: it’s not just about cutting code or reconfiguring CPUs. For example, Queensland is now home to Australia’s only
interactive games cluster. Exciting emerging local areas include e-learning, e-security, e-health and bioinformatics.

The ICT industry increasingly wants people who have a combination of business, problem-solving and interpersonal skills and people who can communicate, who can sell and market and who can develop relationships of trust with customers, suppliers, business partners and within teams. IT skills are readily transferable from one employer to another, and if you want to travel or have a family, the hours are often flexible and involve using mobile technology.

**Why ICT at UQ?**

UQ’s IT program is not only based on the international standard for Computer Science curricula, we helped to write the standard. And now, we’ve added new majors and double-majors in response to industry feedback.

Our programming introduction is based on that of the internationally-renowned Massachusetts Institute of Technology.

At UQ, we go further than just preparing you for your first job. We give you the tools, techniques and theory to understand the many facets of IT and to respond to ongoing industry change. Business futurists predict that jobs based on learning to program to specifications are moving to countries where wages are lower. UQ focuses on the work that is more likely to stay in Australia, including specification and design.

Our programs involve easy-to-follow majors, a host of dual degrees (to combine IT with other study areas) and the flexibility to transfer between IT and Engineering in second year.

UQ is not all theory and no practice. Each of our programs has a strong component of project work and in your final year, you have the option of an industry project to allow you to work for several months in a real work place. Employers keep coming back to UQ’s industry projects because of their good experiences with our students.

You can count industry certification courses licensed by major software developers, like Microsoft and Cisco, towards your degree.

UQ’s IT programs are accredited by the Australian Computer Society. UQ participates in a number of activities to encourage greater female participation in IT.

UQ is rated among the top 50 universities in the world (Times Higher Education Supplement, 2006), the top 5 in the Asia-Pacific region (Shanghai Jiao Tong University rankings, 2006), the top 4 in Australia (Uni Melbourne study, 2004) and number 1 in Queensland (all three studies).

UQ is rated in the top band of Australian universities for teaching quality and number 1 in Queensland.
UQ is Queensland’s top-ranked research university. Because our staff are researchers as well as teachers, they pass leading edge methods on to you. Some of the best teaching and resources for new students spring from the research activities undertaken by staff and senior students.

UQ offers great facilities, beautiful campuses, students from all over the world, and international exchange programs.

UQ offers scholarships worth $15,000 to the best students entering first year from school in Australia, and scholarships that pay tuition fees for certain incoming international students.

For more information see ...

http://study.itee.uq.edu.au/