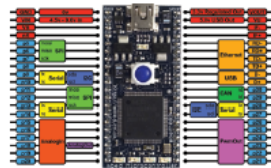




FAST AND EFFECTIVE EMBEDDED SYSTEMS DESIGN

Applying the
ARM mbed



Second Edition

Rob Toulson and Tim Wilmshurst



Chapter 4: Analog Output

tw rev. 26.8.16

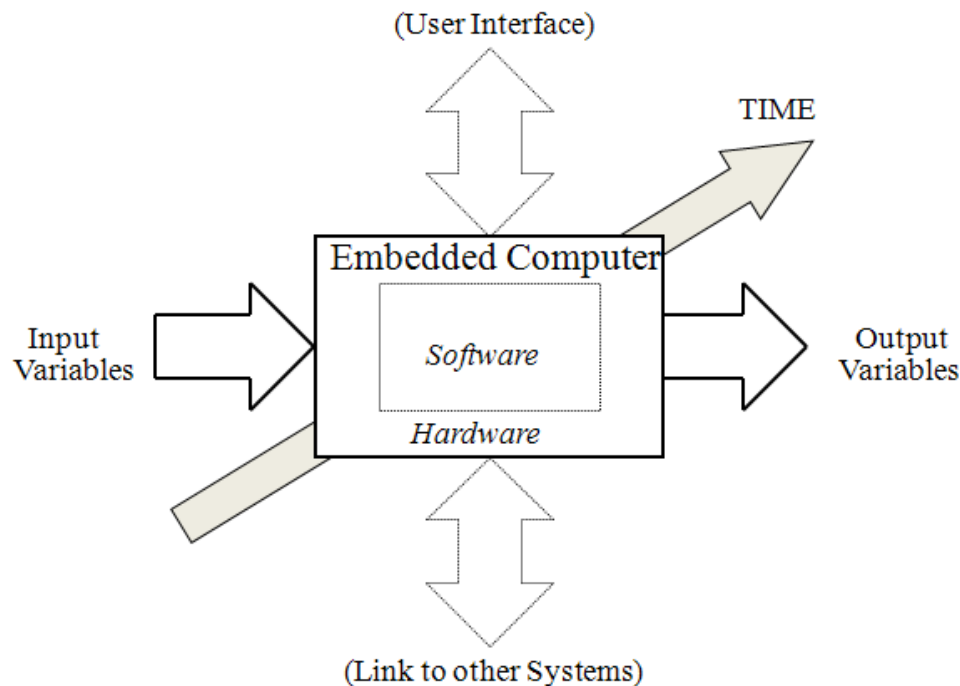
If you use or reference these slides or the associated textbook, please cite the original authors' work as follows:

Toulson, R. & Wilmshurst, T. (2016). Fast and Effective Embedded Systems Design - Applying the ARM mbed (2nd edition), Newnes, Oxford, ISBN: 978-0-08-100880-5.

www.embedded-knowhow.co.uk

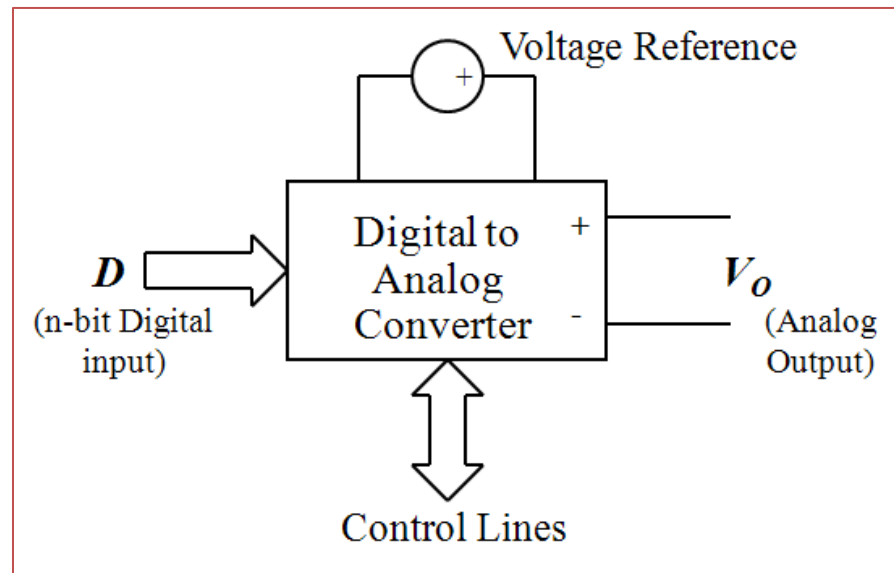
Introducing Data Conversion

Microcontrollers are digital devices; but they spend most of their time dealing with a world which is analog. To make sense of incoming analog signals, for example from a microphone or temperature sensor, they must be able to convert them into digital form. After processing the data, they may then need to convert digital data back to analog form, for example to drive a loudspeaker or dc motor. Here we consider *digital to analog conversion*.



The Digital-to-Analog Converter

A digital-to-analog converter (DAC) converts a binary input number into an analog output. The actual circuitry inside the DAC need not concern us. We can however represent the DAC as a block diagram, as shown. This has a digital input, represented by D , and an analog output, represented by V_o . To work, the DAC needs a *voltage reference* - a precise, stable and known voltage.

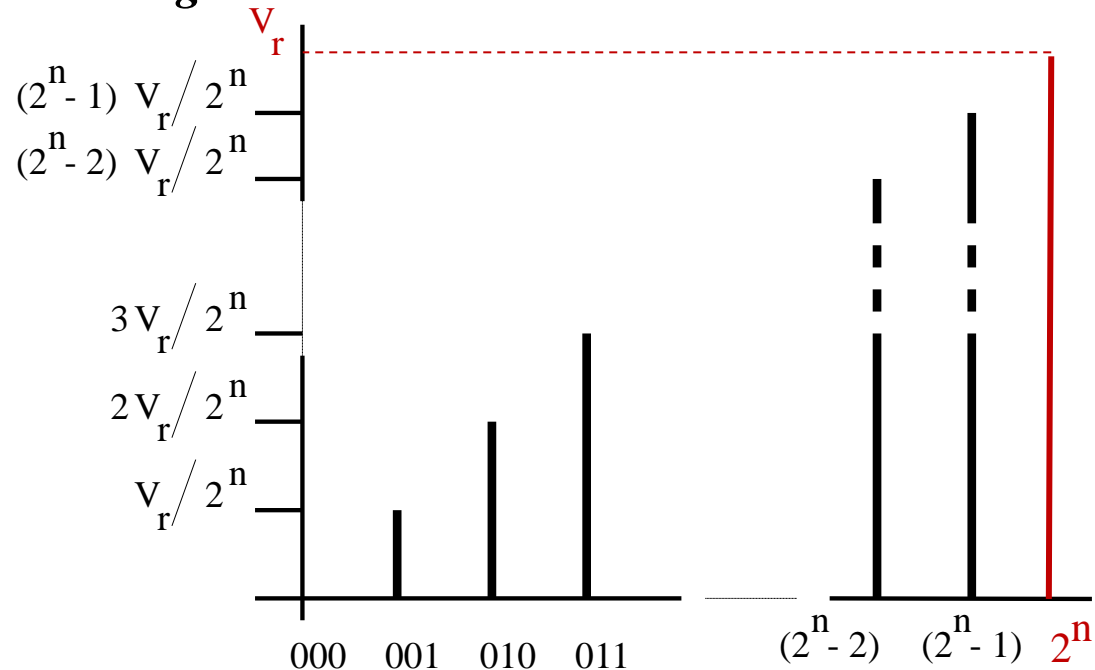


Most DACs have a simple relationship between their digital input and analog output, as shown here. V_r is the value of the voltage reference.

$$V_o = \frac{D}{2^n} V_r$$

DAC Characteristics

Output Voltage

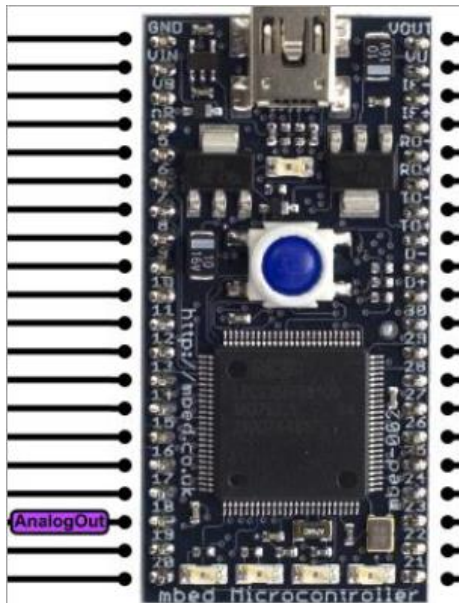


D (Input Binary Number)

The LPC1768 (and hence the mbed) has a 10-bit DAC; there will therefore be 2^{10} steps in its output characteristic, i.e. 1024. Normally it uses its own power supply voltage, i.e. 3.3 V, as voltage reference. The step size, or resolution, will therefore be $3.3/1024$, i.e. 3.22 mV.

Analog Output on the mbed

The mbed has an internal DAC. The output of this connects to the mbed's single analog output, on pin 18. This has a set of API utilities which can be used with it.



Functions	Usage
AnalogOut	Create an AnalogOut object connected to the specified pin
write	Set the output voltage, specified as a percentage (float)
write_u16	Set the output voltage, represented as an unsigned short in the range [0x0, 0xFFFF]
read	Return the current output voltage setting, measured as a percentage (float)
operator=	An operator shorthand for write()

C code
feature

Check your understanding of keywords **float**, **short** and **unsigned**.

Setting Simple Analog Outputs

In this program we create an analog output labelled **Aout**, using **AnalogOut**. It's then possible to set the analog output simply by setting **Aout** to any permissible value. **Aout** takes a floating point number between 0.0 and 1.0. The actual output voltage on pin 18 is between 0 V and 3.3 V; the floating point number is scaled to this.

*/*Program Example 4.1: Three values of DAC are output in turn on Pin 18. Read the output on a DVM.*

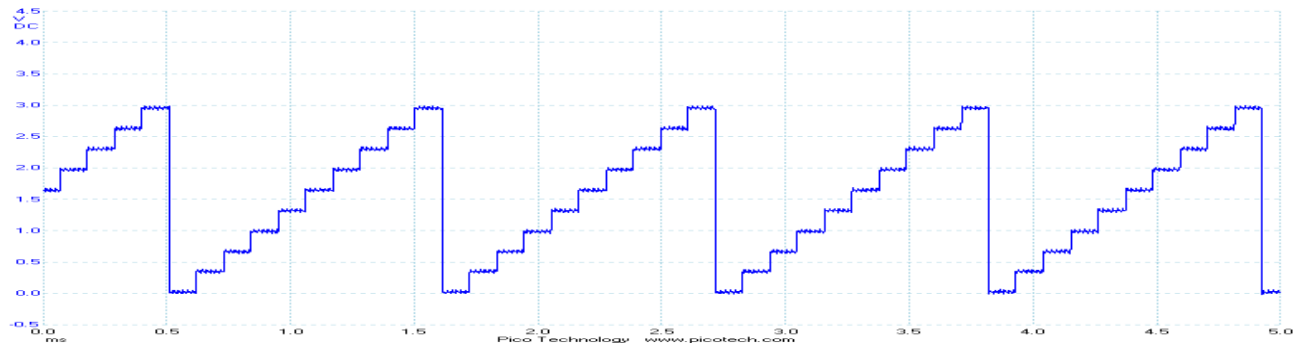
```
*/  
#include "mbed.h"  
AnalogOut Aout(p18);    //create an analog output on pin 18  
int main() {  
    while(1) {  
        Aout=0.25;        // 0.25*3.3V = 0.825V  
        wait(2);  
        Aout=0.5;         // 0.5*3.3V = 1.65V  
        wait(2);  
        Aout=0.75;        // 0.75*3.3V = 2.475V  
        wait(2);  
    }  
}
```

Generating a sawtooth waveform

```
/*Program Example 4.2: Sawtooth waveform on DAC output. View on
oscilloscope */

#include "mbed.h"
AnalogOut Aout(p18);
float i;

int main() {
    while(1){
        for (i=0;i<1;i=i+0.1){           // i is incremented in steps of 0.1
            Aout=i;
            wait(0.001);                  // wait 1 millisecond
        }
    }
}
```



Generating a sine wave

```
/*Program Example 4.3: Sine wave on DAC output. View on oscilloscope
                                                                    */

#include "mbed.h"
AnalogOut Aout(p18);
float i;

int main() {
    while(1) {
        for (i=0;i<2;i=i+0.05) {
            Aout=0.5+0.5*sin(i*3.14159); // Compute the sine value,+ half the range
            wait(.001);                  // Controls the sine wave period
        }
    }
}
```

C code
feature

Here we apply the **sin()** function,
part of the C standard library.

Questions from the Quiz

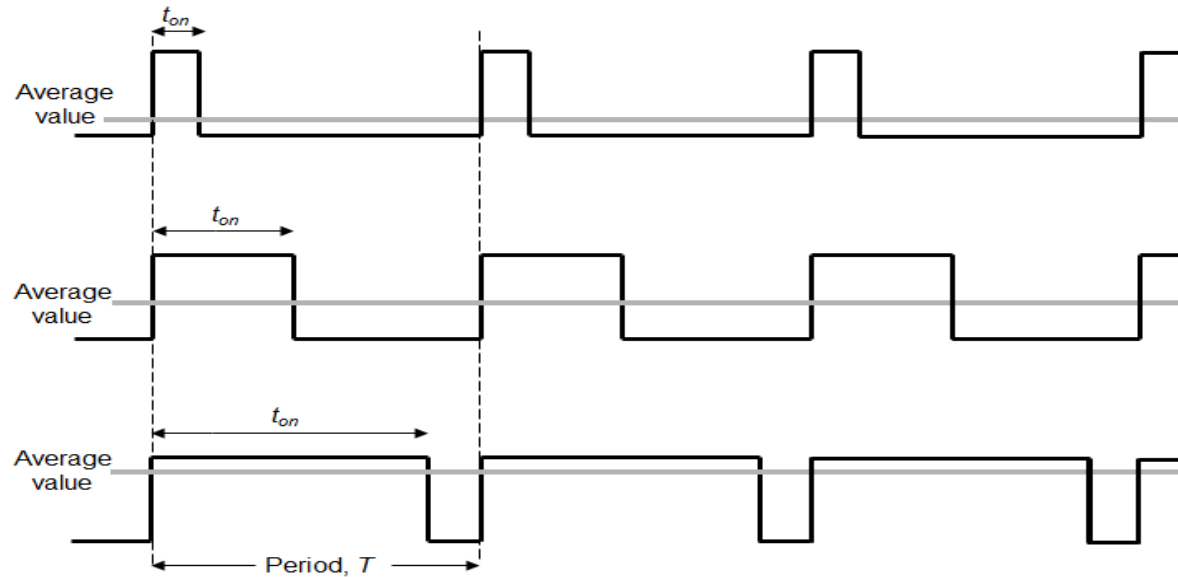
1. A 7-bit DAC obeys the equation shown earlier, and has a voltage reference of 2.56 V.
 - a) What is its resolution?
 - b) What is its output if the input is 100 0101?
 - c) What is its output if the input is 0x2A?
 - d) What is its digital input in decimal and binary if its output reads 0.48 V?
3. What is the output of the LPC1768 DAC, if its input digital word is
 - a) 00 0000 1000
 - b) 0x80
 - c) 10 1000 1000 ?
4. What output voltages will be read on a DVM while this program loop runs on the mbed?

```
while(1){  
    for (i=0;i<1;i=i+0.2){  
        Aout=i;  
        wait(0.1);  
    }  
}
```
5. The program in Question 4 gives a crude saw tooth waveform. What is its period?

Another Form of Analog Output: Pulse Width Modulation

Pulse width modulation (PWM) is an alternative to the DAC; PWM represents a neat and simple way of getting a rectangular digital waveform to control an analog variable, usually voltage or current. PWM is used in a variety of applications, ranging from telecommunications to robotic control.

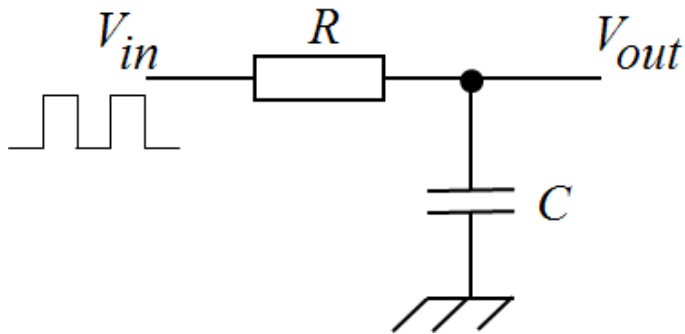
Whatever duty cycle a PWM stream has, there is an average value, as shown. By controlling the duty cycle, we control this average value. When using PWM, it is this average that we're usually interested in.



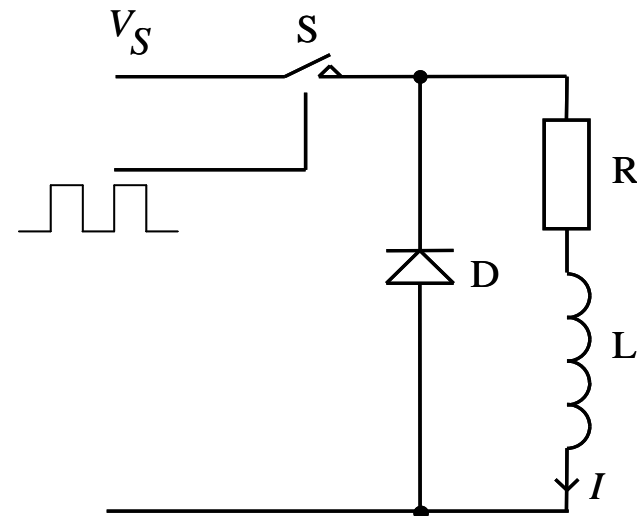
$$\text{duty cycle} = \frac{\text{pulse on time}}{\text{pulse period}} * 100\%$$

Simple Averaging Circuits

The average PWM value can be extracted from the PWM stream in a number of ways. We can use a low-pass filter, like the resistor-capacitor combination below left. In this case, and as long as PWM frequency and values of R and C are well-chosen, V_{out} becomes an analog output, with a bit of ripple, and the combination of PWM and filter acts just like a DAC. Alternatively, if we switch the current flowing in an inductive load, below right, then the inductance has an averaging effect on the current flowing through it. This is important, as the windings of any motor are inductive, so we can use this technique for motor control.



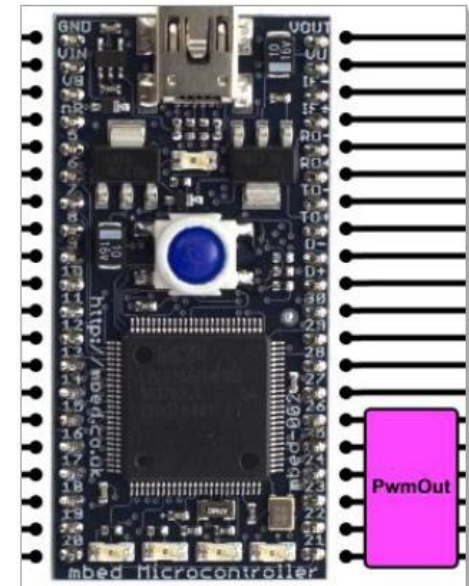
Resistor-capacitor low-pass filter



An inductive load

PWM on the mbed

The mbed has six PWM outputs, on pins 21-26. Each has two variables, period and pulse width, or duty cycle derived from these. Similar to previous examples, a PWM output can be established, named and allocated to a pin using **PwmOut**. All PWM outputs share the same period/frequency.



Functions	Usage
PwmOut	Create a PwmOut object connected to the specified pin
write	Set the output duty-cycle, specified as a normalised float (0.0 – 1.0)
read	Return the current output duty-cycle setting, measured as a normalised float (0.0 – 1.0)
period	Set the PWM period, specified in seconds (float), keeping the duty cycle the same.
period_ms	Set the PWM period, specified in milliseconds (int), keeping the duty cycle the same.
period_us	Set the PWM period, specified in microseconds (int), keeping the duty cycle the same.
pulsewidth	Set the PWM pulse width, specified in seconds (float), keeping the period the same.
pulsewidth_ms	Set the PWM pulse width, specified in milliseconds (int), keeping the period the same.
pulsewidth_us	Set the PWM pulse width, specified microseconds (int), keeping the period the same.
mbed-defined operator =	An operator shorthand for write()

A Trial PWM Output

This program generates a 100 Hz pulse with 50% duty cycle, i.e. a perfect square wave, appearing on pin 21.

```
/*Sets PWM source to fixed frequency and duty cycle. Observe output on
oscilloscope.
*/

#include "mbed.h"
PwmOut PWM1(p21);           //create a PWM output called PWM1 on pin 21
int main() {
    PWM1.period(0.010);      // set PWM period to 10 ms
    PWM1=0.5;                // set duty cycle to 50%
}
```

Speed Control of a Small Motor

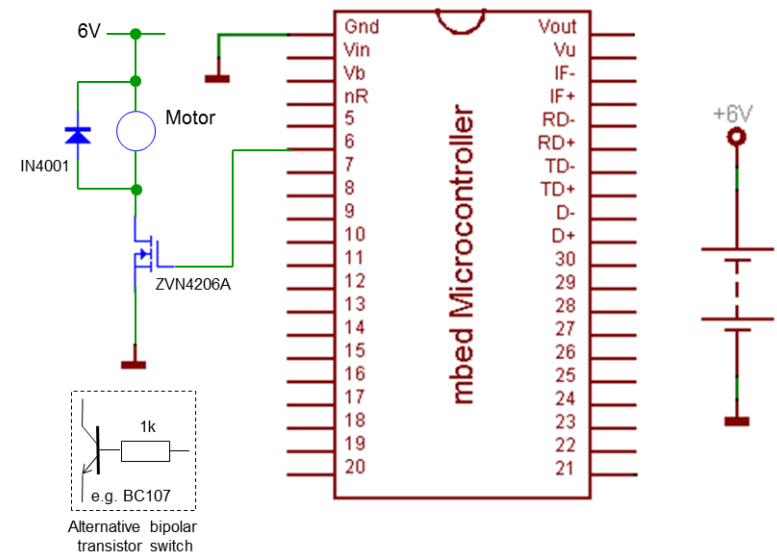
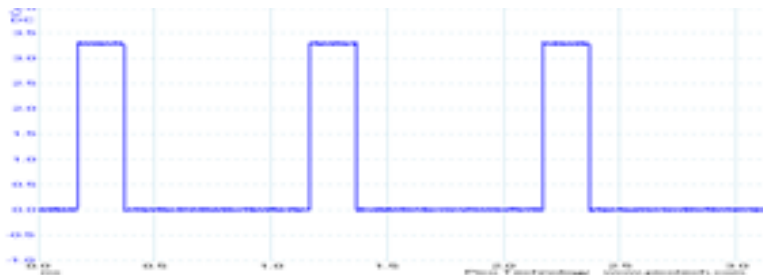
```
/*Program Example 4.5: PWM control to DC motor is repeatedly ramped
*/
```

```
#include "mbed.h"
PwmOut PWM1(p21);
float i;
```

```
int main() {
    PWM1.period(0.010);
    while(1) {
        for (i=0;i<1;i=i+0.01) {
            PWM1=i;
            wait(0.2);
        }
    }
}
```

```
//set PWM period to 10 ms
```

```
//update PWM duty cycle
```



(Gate of MOSFET should now be connected to pin 21)

Generating PWM in Software

Although we have just used PWM sources on the mbed, it is useful to realise that these aren't essential for creating PWM; this can be done just with a digital output, and some timing.

```
/*Program Example 4.6: Software generated PWM. 2 PWM values generated in turn, with  
full on and off included for comparison.
```

```
*/
```

```
#include "mbed.h"  
DigitalOut motor(p6);  
int i;
```

```
int main() {  
    while(1) {  
        motor = 0;                //motor switched off for 5 secs  
        wait (5);  
        for (i=0;i<5000;i=i+1) {  //5000 PWM cycles, low duty cycle  
            motor = 1;  
            wait_us(400);          //output high for 400us  
            motor = 0;  
            wait_us(600);          //output low for 600us  
        }  
        for (i=0;i<5000;i=i+1) {  //5000 PWM cycles, high duty cycle  
            motor = 1;  
            wait_us(800);          //output high for 800us  
            motor = 0;  
            wait_us(200);          //output low for 200us  
        }  
        motor = 1;                //motor switched fully on for 5 secs  
        wait (5);  
    }  
}
```

Questions from the Quiz

7. A PWM data stream has a frequency of 4 kHz, and duty cycle of 25%. What is its pulse width?

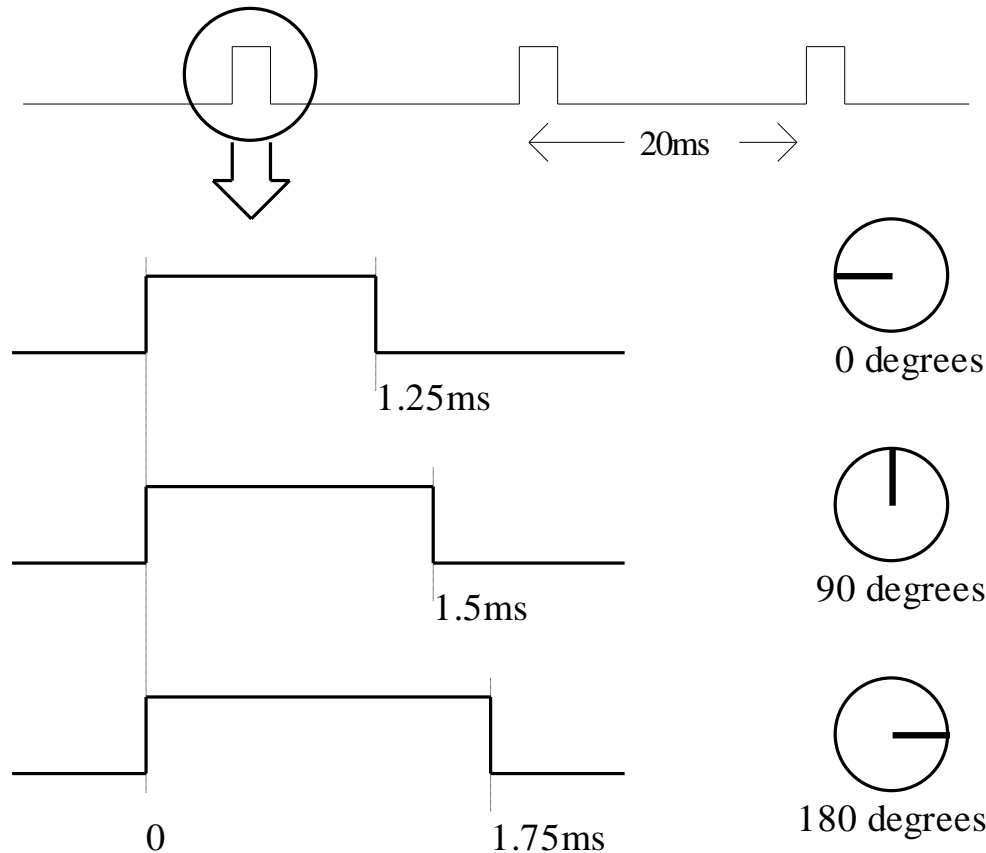
8. A PWM data stream has period of 20 ms, and on time of 1 ms. What is its duty cycle?

9. The PWM on an mbed is set up with these statements. What is the on time of the waveform?

```
PWM1.period(0.004);    // set PWM period
PWM1=0.75;              // set duty cycle
```

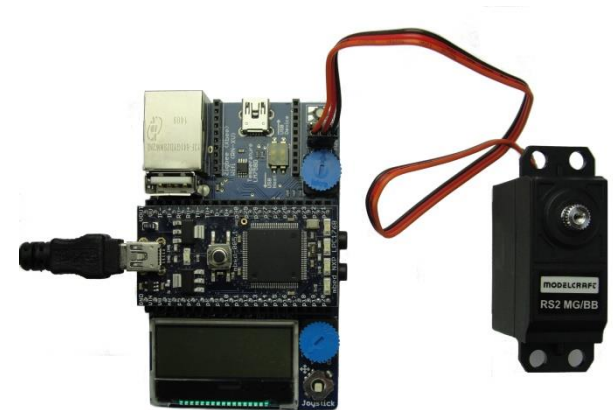
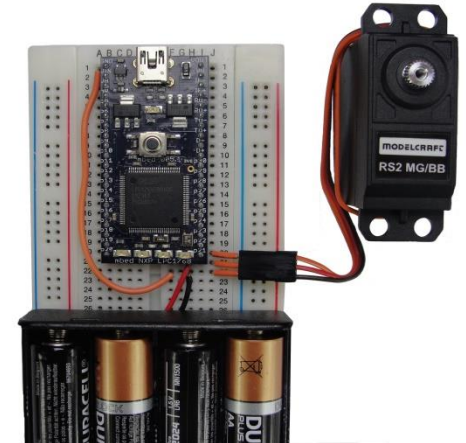
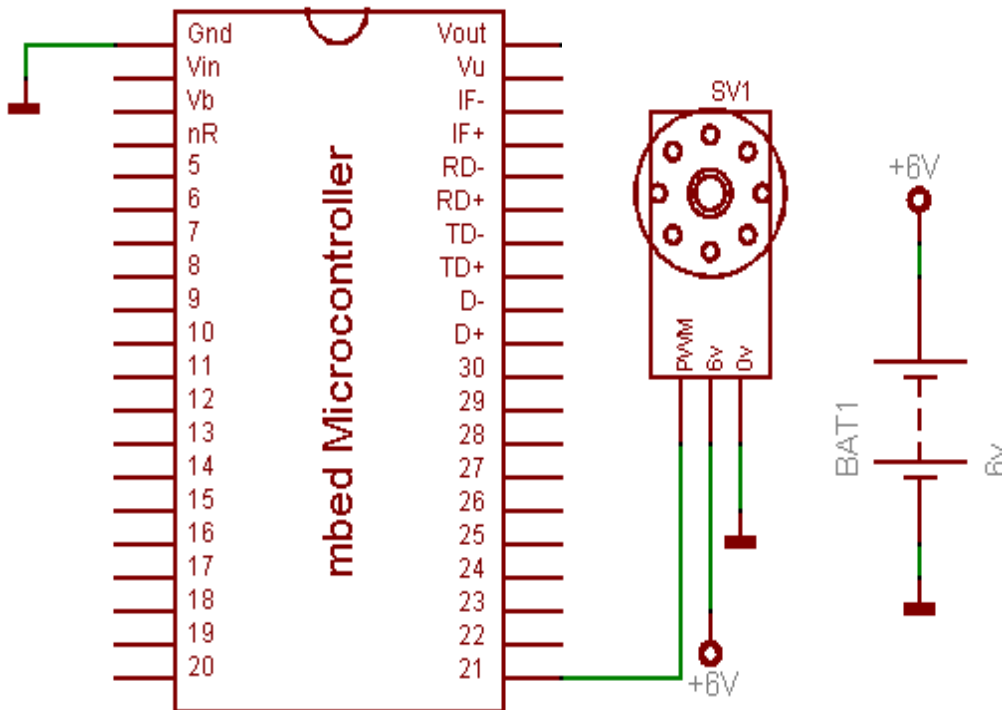
Servo Control

A servo is a small rotary position control device, often used in radio-controlled cars and aircraft to control angular position of variables such as steering, elevators and rudders. Servos are now popular in certain robotic applications. The servo shaft can be positioned to specific angular positions by sending the servo a PWM signal, as shown.



Servo Control with the mbed

The servo requires a higher current than the USB standard can provide, and so it must be powered using an external supply, e.g. with a 4xAA (6 V) battery pack. The app board has PWM output connectors, so both breadboard or app board builds are possible.



Producing Audio Output

```
/*Program Example 4.7: Plays the tune "Oranges and Lemons" on a piezo  
buzzer, using PWM  
*/
```

```
#include "mbed.h"  
PwmOut buzzer(p21);
```



```
float frequency[]={659,554,659,554,440,494,554,587,494,659,554,440}; //frequency array  
float beat[]={1,1,1,1,1,0.5,0.5,1,1,1,1,2}; //beat array  
int main() {  
    while (1) {  
        for (int i=0;i<=11;i++) {  
            buzzer.period(1/(2*frequency[i])); // set PWM period  
            buzzer=0.5; // set duty cycle  
            wait(0.4*beat[i]); // hold for beat period  
        }  
    }  
}
```

C code feature

The program uses two arrays, one defined for frequency data, the other for beat length. There are 12 values in each, for each of the 12 notes in the tune. From the frequency array the PWM period is calculated and set, always with a 50% duty ratio.

Chapter review

- A digital-to-analog converter (DAC) converts an input binary number to an output analog voltage, which is proportional to that input number.
- DACs are widely used to create continuously varying voltages, for example to generate analog waveforms.
- The mbed has a single DAC, and associated set of library functions.
- Pulse Width Modulation (PWM) provides a way of controlling certain analog quantities, by varying the pulse width of a fixed frequency rectangular waveform.
- PWM is widely used for controlling flow of electrical power, for example led brightness or motor control.
- The mbed has six possible PWM outputs. They can all be individually controlled, but must all share the same frequency.

