# Embedded systems Projects

## Administration and Evaluation

You will need to submit a formal report concerning your project work. The report should be approximately 4,000 words in length, and should be submitted to the Teaching Office on the first teaching day of the Lent Term. Students are also encouraged to submit their lab books.

The report will normally contain sections detailing the problem being addressed, the options considered for solving the problem, the calculations made to ensure that the proposed solution can meet the technical limits and requirements (for example memory capacites, limits of throughput on connections), and the way in which it is proposed to test to show that the solution does in fact work.

Since some of these projects have not been tried before, it may well be that the eventual solution is different from the one originally envisaged. If this is the case, then add a section explaining the difficulty encountered with the original approach to the problem, and details of the workaround or improvement made.

## Support

Many of the projects use the same basic functionality. We have made a Printed Circuit Board [PCB] available, which may help you. Details are available here: http://www.cl.cam.ac.uk/teaching/1011/P31/project/project_pcb.pdf

You may have ideas for a project other than the ones on the project_suggestions list. If so, talk to Dr Wassell or Brian Jones early on, to make sure both that the project is suitable, and that we can get hold os any parts required. There is a huge range of sensors available, many of which produce a voltage, and so are easy to interface to a microcontroller, plus a large number of sensors and peripherals use interconnections for which there is support built into the microcontroller, for example I2C bus or SPI bus.

To make writing your report easier, and also to make it easier for the demonstrators to help you with your project, keep paper or electronic copies of relevant datasheets, and keep notes in a lab book of key items which are relevant to the project, such as any unusual properties for the parts you are using.

## General project advice

Choice of microcontroller

There are basically 4 choices which we have available in a Dual In Line (DIL) package. (There are lots more available in surface mount packages, but they require Printed Circuit Boards rather than the prototyping boards we have). All 4 devices have a built in temerature sensor, and all three are available in a 'V' version with a lower maximum clock speed but the ability to work at supply voltages down to 1.8V instead of 2.7V All 4 devices have much the same functionality available. Data sheets for each are in the docs folder.

- ATMEGA168 - the one you have been using. 1K of ram, 1 USART

  The makefile settings for the MCU (line2 of the Makefile) and the avrdude programming are:
  `MCU=atmega168 avrdude -p m168p`

- ATMEGA328 - the one you have been using, but with 2K of ram, 1 USART

  The makefile settings for the MCU (line2 of the Makefile) and the avrdude programming are:
  `MCU=atmega328 avrdude -p m328p`

- ATTINY45 - a physically small, 8 pin device, with much the same functionality as the ATMEGA168, but usually the pin count . Only 256 bytes of ram on this device. You might want to use this one

where space is at a premium, or where you are using it as a sensor. When using this device it may be necessary to move it to a programmer rather than program it in place.

The makefile settings for the MCU (line2 of the Makefile) and the avrdude programming are:
`MCU=attiny45 avrdude -p t45`

- ATMEGA644 - 2 USARTs, 4K of RAM, physically large (50 mm long x 18 mm wide). GPS and serial interfacing projects may need to use 2 USARTS. Data logging projects which write large amounts of data to SD cards will need the RAM capacity.

  The makefile settings for the MCU (line2 of the Makefile) and the avrdude programming are:
  `MCU=atmega644p avrdude -F -p m644p`

Choice of speed

If the project will eventually be battery powered, choose a frequency for the microcontroller within the range suitable for the battery voltage. You can always test on the bench using 5V, but changing the operating speed part way through the project is likely to cause problems. In general you should choose top operate the CPU as fast as possible, unless you are really trying to minimise the current drawn. It uses the same power to finish the processing in half the time and go into sleep mode, as to run at half the CPU speed.

Storage

If you can process your data on the microcontroller, do that. If you need to store up to 2 Mbytes, you can add flash memory to the SPI bus, but will need to write a 256 bytes page at a time. If you need to store up to 2 Gbytes, then you can use an external MicroSD card, but will need to use a microcontroller with at least 2K RAM, plus you need to provide a function to provide a timestamp for your files (you can either add a Real Time Clock (RTC) or provide a dummy function to write a plausible timestamp.

Libraries

There is software support in the form of example implementations or libraries for the following devices:

- DS1302 Real Time Clock http://www.cl.cam.ac.uk/teaching/1011/P31/lib/ds1302.c
- M25P16 SPI Flash memory http://www.cl.cam.ac.uk/teaching/1011/P31/lib/m25P16.c requires SPI support http://www.cl.cam.ac.uk/teaching/1011/P31/lib/spi2.c
- MicroSD memory card and FAT 16 filesystem http://www.cl.cam.ac.uk/teaching/1011/P31/lib/ff.c Requires a real time clock (or dummy functions), and SPI support http://www.cl.cam.ac.uk/teaching/1011/P31/lib/spi2.c
- SCA3000-E01 3 axis accelerometer http://www.cl.cam.ac.uk/teaching/1011/P31/lib/accelerometer.c requires SPI support http://www.cl.cam.ac.uk/teaching/1011/P31/lib/spi.c

Soldering a row of 5 pins in to JP3/4 and connecting 2->3 with a shunt will form a loopback (echo) connection.

Serial

You have used serial communications using the RS232 protocol, using the serial_to_USB converters. If you need to use the COM1 ports on the PC instead, then you need to know that they are device /dev/ttyS0 in Linux. They are there permanently - they do not appear and disappear on plug in/out like USB devices. You will find on first use that you don't have access to /dev/ttyS0. To fix this type sudo chown <crsid> /dev/ttyS0 replacing <crsid> with your crsid. You will be asked for your login password - this is the Operating system's way of getting confimation that you want to run a priveleged command (in this case chown)

If you are using the level converter PCB, then, once connected to the PC you can look at JP3 and JP4 pin3 to see the incoming signal from the PC. Typing characters into minicom with the port set to /dev/ttyS0 should be visible with an oscilloscope.

6 pin programming header

The Tuxgraphics programmers you have used so far have a 5 pin connector. On some of the Printed Circuit boards, you may see a 6 way connector, whose pin connections are as follows: `GND Reset (+5V) rarely used or connected SCK MISO MOSI` I will provide adaptors where necessary

Devices

We buy most of our devices from Farnell (onecall.farnell.com) or RS components (rswww.com) If the part or the bag it is in has a 6 or 7 digit number, it is very likely to be either the Farnell or RS part number and you can use the website to look up data for the part.

Development

During the planning stage, work out a series of steps to get to the eventual goal, and how you will test each stage. Take notes as you go, which should help you when writing the report at the end of the project.

Work incrementally, testing as you go, or you run the risk of getting stuck at a complex part of the project, and after considerable time discovering that the fault was in something simple from an earlier stage which is not working as expected.

Even if the project will eventually be battery powered and mobile, do as much development and testing as possible in the lab environment, and only go mobile when you can do no more in the lab.

You should consider using several computers to speed your development - develop, read code and program your device from one, and have minicom or other test programs ready to go on another. Remember that you can be logged in to multiple computers, and that you can have multiple terminal windows open on each.

When developing, as you get to a stage where something is working, add a brief comment to the file, and store a copy. That way you can go back to a working version later.

# Notes on power consumption

1. Sleep. Spend as much time as possible in the lowest power sleep mode. Beware though, of turning off the clock which is running the wake-up source.

2. Turn off parts of the microcontroller which aren't in use. As above, beware turning off the clock which runs the timer which you need to wake. There are no warnings to tell you that you have done this.

3. Don't let inputs float. Pull them up, or turn them into outputs, and assign them to be high or low depending whether the load is attached to the supply or ground respectively.

4. Beware of switches on inputs. As inputs the pullups will cost power if the switch is closed. If you aren't using a switch, make the pin an output and set it low. Alternatively, choose the default position of the switch to be all open.

5. The eye will detect a 20mS flash on an LED, so you can keep flashes to this length to save power. LEDs need a few mA to light up.

6. On the project PCB, if you aren't using the voltage reference IC2, remove the link to the positive supply, as the reference needs 210uA to work.

7. Also on the project PCB, be aware that if C1 is fitted it will take time to charge through the resistor, which will disrupt any current measurements you make.

# Common pitfalls

Here is a list of common pitfalls, in no particular order.

## Speed vs power and working voltage

The microcontroller uses much less power at 3.3 Volts than at 5 Volts, and less still at lower voltages, but beware that below 2.7 Volts it cannot run at full speed.

If your project needs to be battery powered, use the Lithium batteries, and run the micro either at battery voltage, or at 3.3 Volts through a voltage regulator. A voltage regulator costs 17uA of idle current, but the reduction in consumption of the other circuitry probably saves you more.

## Interrupts

One of the easiest traps to fall into is to spend so long in the interrupt routine, that another interrupt has occurred before the first one is finished. If timing is critcal, then the extra latency before the second interrupt is serviced will probably cause a failure. This suggests doing the minimum possible in the interrupt service routine (ISR), eg setting flags, and the maximum outside the ISR.

A different strategy to the above is as follows:

Do everything in the interrupt routines, and nothing in main. Make sure that each interrupt handler sets up the conditions for the next interrupt to occur. If only one interrupt is enabled at a time, then there is no potential to clash.

Make sure that all interrupts are set up and conditions cleared before the Master Interrupt Enable bit is set, otherwise an interrupt may occur immediately.

If you use global variables which are changed in more than one place, declare such variables as `volatile`, otherwise the compiler will optimise the code, and not notice that the variable has changed.

## I/O

It is very easy when modifying code to forget to define a new output. Everything will compile correctly, but the pin will not behave as you expect.

It is useful to copy a line such as `PORTB |= (1<<PB1)`, changing it to `PORTB &= ~(1<<PB1)`. However, omitting the brackets in the second case will cause problems because of operator precedence.

Except while initialising, it is *very* rare that you will need assignment such as `PORTB = (1<<PB2);` You almost always need `|=` or `&=`

## Analogue to Digital converter

The most common pitfall with the A to D is forgetting to supply a reference volatge at AVcc. Zero or very low readings are a symptom of this.

The next most common is having too high a clock rate. If it is too fast then only the first few bits of the data will be correct.

Selecting the wrong analogue input is the next most common error.

## Fuse errors

If you can no longer program the fuses, have you disabled the clock by mistake ? This is particularly easy to do when chaning from an internal to an external or crystal derived clock.

If you reprogram the fuses, the device does not automatically restart. You must power cycle it. Changing the program *does* however force a restart.