# Embedded System Programing Lab4

In this lab we are going to perform following task:

- Installing avr-libc-1.4.3
- Understand avrlib
- Perform Lab3 experiment using avrlib

At the end of lab you have to submit soft copy of your experiment report. In this report you will include the following points:

- Experiment setup:
- Brief Circuit Description if any
- Which Registers you had used and Value of each register with short explanation why you had set that value.
- Results
- Output port results
- Reading or Observation of external devices (Sensor, LEDs)
- Problems you had faced in experiment and how you have solved them.

**Task 1: Installing** avr-libc-1.4.3 **Library**

Download zip file from ESP(current)->Embedded System Lab. Using *tar, make* & *make install* utility install avr-libc-1.4.3 library. In your report, include brief description of command and options that you had used and directory in which the library has been installed.

**Task 2: Understand avrlib Library**

AVRlib is an open-source collection of C-language function libraries for the Atmel AVR series processors. The goal of AVRlib is to provide the programmer with a code base which performs the most often needed tasks in embedded system programming. Hopefully, this will allow the programmer to focus on high-level operation of their code rather than get bogged down in the details of low-level code.

In short, AVRlib is a bunch of functions that do things commonly needed in embedded systems. Despite the learning curve of getting started, for most projects, using AVRlib will shorten the time spent programming and/or improve the quality or functionality of the final product.

AVRlib functions are available for a wide variety of tasks and purposes. In general, AVRlib tries to address the following kinds of needs:

- Functions which control or interface to AVR processor hardware (like timers, uarts, the a2d converter, etc)
- Functions which interface to or drive devices often used in embedded systems (like LCDs, hard disks, gps units, etc)
- Functions which create higher-level functionality from processor resources (like pulse generation, software uarts, software i2c bus, etc)

Download zip file from ESP(current)->Embedded System Lab. Using unzip. Unzip this file using unzip utility. In your report include brief description of command that you had used with options.

You should now have an **avrlib** directory where you installed AVRlib. If you have some time, get familiar with what's inside some of the directories. Your directories should look something like this:

avrlib <-- AVRlib header and code files
avrlib/conf <-- AVRlib template configuration files
avrlib/docs <-- AVRlib documentation
avrlib/examples <-- AVRlib example applications
avrlib/make <-- AVRlib makefile include (avrproj_make file in here)

Finally, you need to create an environment variable **AVRLIB** which points to the directory where you "installed" or unzipped the AVRlib files so the compiler can find them. An example might be:

**$export AVRLIB =/opt/avr/avrlib**

Now change directories to the location where you installed AVRlib.
For example:
**$cd /opt/avr/avrlib**

Go into the examples directory.
$**cd examples**

Pick an example to try compiling such as rprintf and change to that directory. cd rprintf
Type **make clean** at the prompt
Type **make**

If your output looked like this then you just compiled your first AVRlib program:

.....
.....

| text | data | bss | dec | hex filename |
|---|---|---|---|---|
| 9596 | 0 | 192 | 9788 | 263c rprintftest.elf |

Errors: none
rm /opt/avr/avrlib/vt100.o /opt/avr/avrlib/rprintf.o /opt/avr/avrlib/ua
rt.o /opt/avr/avrlib/timer.o /opt/avr/avrlib/buffer.o

In your report write you results and possible resign for the it.

### Task 3: Perform Lab3:Task3 using avrlib

In last lab we had done mini project, in which we read light intensity from LDR using ADC and changed the motors speed accordingly using PWM. ***Avrlib*** provides ready to use code for serial communication, analog to digital conversion and pulse width modulation in file uart.c, a2d.c and timer.c respectively. Test program are also given in *example* directories.

1. First execute this given examples and record output of it and then try to change that output with different parameter setting.

2. Next perform Task3 of Lab3 using avrlib functions.

For that you have to create a new directories in $AVRLIB/example directories.
Copy global.h and makefile in newly created directories.
Create you c life that will use, *uart.h, a2d.h and timer.h* and execute the give task.

Give list of important environment variable and global program variables with their values. Also include What alternate values are possible for them.