University of British Columbia
Electrical and Computer Engineering
ELEC292

Module 4 – Photoelectric Heart Rate Monitor

Introduction

Embedded systems are often designed to perform simple or repetitive tasks while connected to larger computers. Examples of such systems are found in many of today's computers: the mouse, keyboard, memory sticks, hard drive controllers, etc. For this module, you will design, build, program, and test a microcontroller-based photoelectric heart rate monitor. This kind of heart monitor is attached to a person's finger to measure the heart rate. The device you will be building is based on a non-invasive photo transmitter/receiver pair sensor which detects small variations of blood flow in the fingers due to the heart pumping action. After filtering and amplifying such variations, a microcontroller-useable signal can be generated. The heart rate monitor will serially transmit the obtained data to a personal computer using the serial port. You will program the personal computer using Python to receive the heart rate and conveniently present it in real time using a strip chart plot.

There are many free python distributions available. One that has all the functionality to complete this laboratory module is WinPython version 3 available at:

https://winpython.github.io/

In this module you will be programming the AT89LP51RC2 microcontroller using the C programming language. C is the de facto standard for the programming of embedded systems.

Since the amount of work required to complete this module is more significant than in previous modules, you can work with a partner and submit just one circuit and program for both of you.

References

C51 user manual included with the latest version of CrossIDE.

LM335 data sheet

AT89LP51RC2 reference manual.

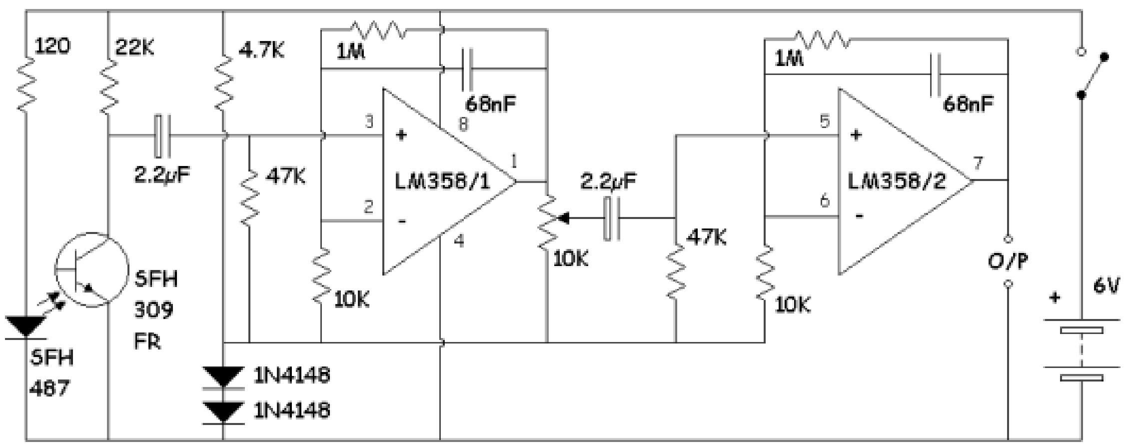
Python reference manual. Available online.

Amplifier and Filter

There is some information about photoelectric heart rate sensors in the internet. A good article with a single power supply amplifier/filter can be found at:

http://www.picotech.com/experiments/calculating_heart_rate/

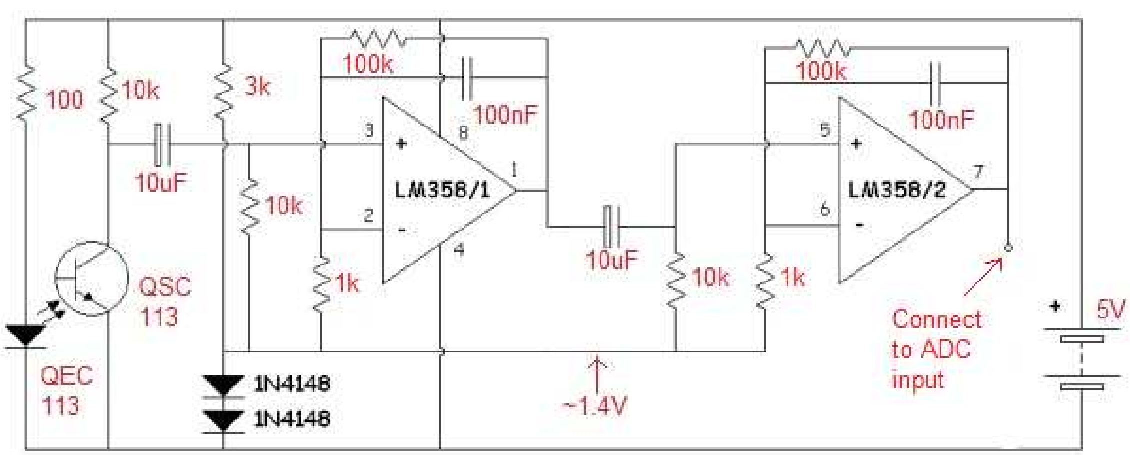
The circuit below comes from that above.



If you try to duplicate the circuit above you may notice that some of the components needed are not available in your parts kit.

Modified Circuit

Since the component values shown in the circuit above may not be available in your parts kit, we need to replace them with components you have. The circuit below shows the changes (in red) that seem to work satisfactorily:



Notice that the modified circuit is using a 5V voltage source and that the 10k potentiometer has been removed.

Laboratory

1. Build a photo-sensor finger-clip. You can use the instructions available in the web page for the course. This is just a suggestion. You can build your finger clip in any way you want, as long as it works. All the parts required to build the suggested finger clip will be provided in the lab, including the 3mm infrared LEDs (DigiKey part number QEC113-ND) and 3mm infrared phototransistors (DigiKey part number QSC113-ND).

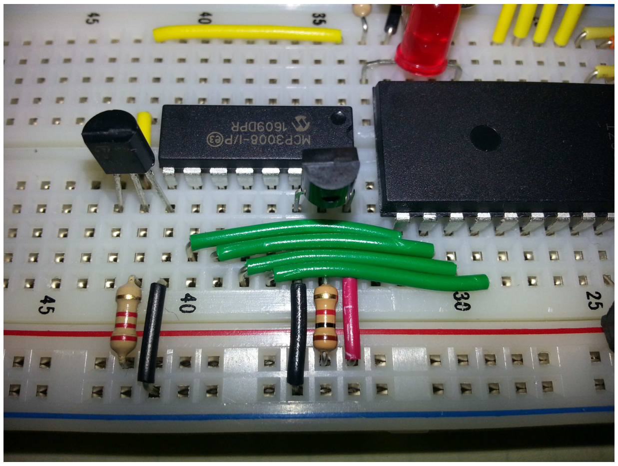
2. Assemble the amplifier/filter circuit from the section above.

3. **Testing C with the AT89LP51RC2.** The program below prints "Hello, World!" in PUTTY running in a computer throughout the serial port of the AT89LP51RC2. Copy and paste it to Crosside and save it to 'hello.c'. To compile it under Crosside with C51, click 'Build' → 'Compile/link with C51'. Make sure you set the 'Complete path to C51.exe' correctly. After compiling, load the resulting '.hex' file to the AT89LP51RC2 microcontroller.

```
//  AT89LP51RC2 "Hello, World!" example.
//  ~C51~

#include <stdio.h>
#include <at89lp51rd2.h>

#define CLK    22118400L // SYSCLK frequency in Hz
#define BAUD     115200L // Baud rate of UART in bps
#if ((CLK/(16L*BAUD))>0x100L)
#error "Can not set baud rate because (CLK/(16*BAUD)) > 0x100 "
#endif
#define BRG_VAL (0x100-(CLK/(16L*BAUD)))

unsigned char _c51_external_startup(void)
{
    AUXR=0B_0001_0001; // 1152 bytes of internal XDATA, P4.4 is I/O

    // Configure serial port and baud rate
    PCON|=0x80;
    SCON = 0x52;
    BDRCON=0;
    BRL=BRG_VAL;
    BDRCON=BRR|TBCK|RBCK|SPD;

    return 0;
}

void main (void)
{
    printf("Hello, World!\n");
}
```

PuTTY is a free Telnet/SSH/Serial terminal that can be downloaded from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. It is possible to launch PuTTY directly from CrossIDE by pressing <Control>+T. Configure PuTTY to 115200 baud, 8 bits, parity none, 2 stop bits, and Flow Control 'none'. Also make sure the 'Complete path of PuTTY.exe' field points to a valid location. Press the reset button of the AT89LP51RC2 to see "Hello, World!" printed in PuTTY.

4. **ADC Converter.** Available on Canvas is the test program 'ADC_SPI.c'. This program reads the analog inputs of an external ADC using SPI. The ADC available this year is the MCP3008 made by Microchip. Compile, load, and test the program. The picture below shows a LM335 temperature sensor output connected to analog input '0' of the MCP3008. Although using the LM335 for this lab is not required, it can be used to quickly test the functionality of the MCP3008 ADC: the voltage measured from analog input '0' should be very close to 3V.



5. **Using Python to communicate with the AT89LP51RC2 Microcontroller.** The Python script shown below opens the serial port in the host computer, constantly reads and prints a received value, and finally closes the serial port when CTRL+C is pressed in Python's command console. Attach the output of the amplifier circuit of point 2 to one ADC input of the MCP3008. Modify the program from the previous point so it transmits the measured value through the serial port every 100 mili-seconds.

```
import time
import serial

# configure the serial port
ser = serial.Serial(
    port='COM1',
    baudrate=115200,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_TWO,
    bytesize=serial.EIGHTBITS
)
ser.isOpen()

while 1 :
    strin = ser.readline()
    print (strin)
```

The script above assumes you are using COM1. For other serial ports, adjust accordingly. Also, Python expects a new line escape sequence ('\n') for each received value from the microcontroller ('\r' will not work).

6. **Heart Rate strip-chart using Python.** The script 'stripchart_sinewave.py' shows how to implement strip-charts in Python. A strip-chart can be used to plot the heart rate transmitted from the AT89LP51RC2 microcontroller board to Python in real time. Modify the provided script so it plots the data received from the serial port. Don't forget to add extra functionality and/or features for bonus marks! Upload to canvas:

   a) Python code.

   b) C code.

   c) ONE good resolution picture of the microcontroller system with the amplifier and finger sensor attached.

   d) A video showing the heart rate strip chart working.

   Only one submission is needed from a team of two students. Please indicate with your submission the name and student number of the team members.

   NOTE: The top mark for this lab is 85% when all the requirements are completed. This is an automatic 10% bonus mark compared with other laboratory assignments.