Lesson 6
Programming Arduino Example 9.7

Multitasking Environment

- A multitasking environment consists of multiple processes, tasks or threads.
- Consider Example 9.1.
- A set of Functions void north_south_Green (), and void east_west_Red () can be considered as task_1.
- A set of functions void north_south_Red (), east_west_Green () can be considered as task_2, which executes next.

Multitasking

- The tasks runs under OS control (supervision) and can do the scheduling.
- Advantage of Multitasking is that when any one task blocks, for example, north-south pathway lights need to remain in same state for some period, and needs to wait for next state then the OS can save the required parameters of that task and load the parameters of some other task and run that.

Multithreaded Environment

- Consider Example 9.2 and an alternative.
- A set of Functions void north_south_Green (), void north_south_Yellow () and and void north_south_Red () can be considered as three threads thread_1, thread_2 and thread_3.
- The thread_1 executes, then blocks for a period, then thread_2 executes then blocks for a period, then thread_3 executes, ……..

Multithreading

- The thread also runs under OS control (supervision) and OS can do the scheduling.
- Advantage of multithreading is that when any one blocks, for example, north-south pathway lights need to remain in same state for 30s, and needs to wait for next state then the OS can save the required parameters of that task and load the parameters of next thread and run that.

The OS threads

- Can run in sequences when each thread has normal priority.
- OS can also be made to run such threads for a specified time slice in each cycle of thread run sequences.
- The OS can assign priorities, make certain thread(s) priority higher than another, to enable certain tasks to execute in real time. y oy

Multitasking and Multithreading

- Each task has independent memory block assigned for that. Two or more threads can share some part of a memory stack and may correspond to a common task.
- The threads and tasks, both runs under OS control (supervision).
- Certain OSes provide for just multithreading and certain both multiple tasks and task consisting of several threads.

LEDs at 12 ports connected for four pathways traffic-lights

- Each pathway has red, yellow and green lights represented by three LEDs ledR, ledY and ledG for each of the 4 pathways
- 12 Variables used in the declaration program, ledR0, ledY0 and ledG0, ledR1, ledY1 and ledG1, ledR2, ledY2 and ledG2 and ledR3, ledY3 and ledG3
- Computer connects to the board using serial pins

Example 9.7

- A complete program for switching ON and OFF of north and south pathways and east and west paths ways, after intervals of 10 s each.
- The green and red LEDs remain in ON states or OFF for 30 s period. Function delay(30) provides the intervals of 30 s
- The delay (10) provides intervals of 10 s between switching ON of traffic along north and south pathways and switching ON of traffic along east and west pathways. Yellow LEDs switch ON for 10s before they switch OFF.

Example 9.7 Step 1 in preprocessor commands

- /* include Posix Thread library functions in Linux distribution or include functions of OS being used */
- # include <pthread.h> //
- // declare thread delete request false to disable delete of the thread(s) bool delete_request false
- // Define a class, named LightsThread for Traffic Lights Project

Example 9.7 Step 2 for declaring class for a thread and creation of threads

- class SeqThread : public PThread {
- }
- /* Assign threadID number to this SeqThread*/
- SeqThread : : SeqThread (int threadID) {
- this -> threadID = threadID;

Example 9.7 Step 3 for setup ( )

- /* Each thread has ID from 1 to 9.*/
- /*declare number of sequential running threads */

  int numseqthread = 9;
- Creating a main_thread_list consisting of numseqthread which sequentially execute
- Write statements in each thread
- /*Write statements for the eight SeqThread objects of class SeqThread for the functions in Examples 9.1 and 9.2*/

Example 9.7 Step 3 for setup ( )

- // SeqThread (1) waits of signal 1 and runs Function north_south_Green () and Function east_west_Red () */

loop ( )

- bool SeqThread :: loop () {
- /* OS runs all nine threads at the main_thread_list as long as loop return false
- Boolean variable is true*/,
- // check for delete_request true

Summary

We learnt

- Programming using threads
- Programming using OS functions for the delay and signal ( )
- Multithreading enables control of the program effeciently, the running of each thread at the CPU, saving of the required thread parameters and CPU running other thread when one blocks

End of Lesson 6 on
Programming Arduino Example 9.7