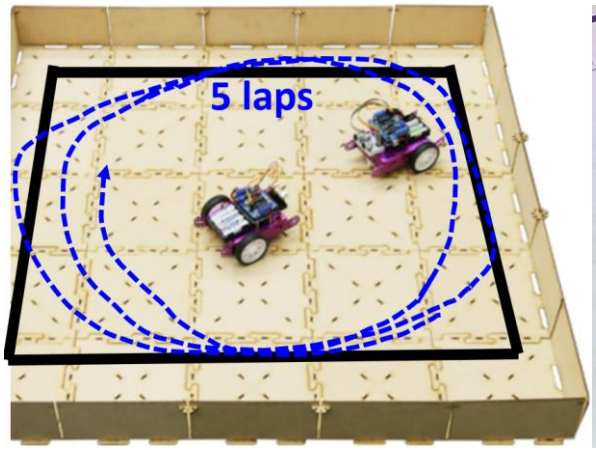
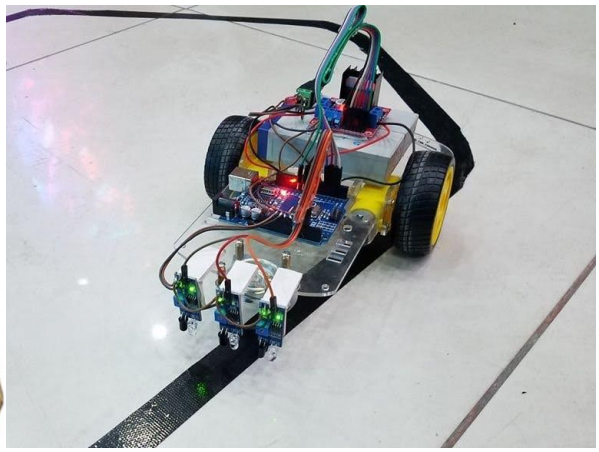
COSE421 Embedded Systems

Class Project

No late turn-in accepted

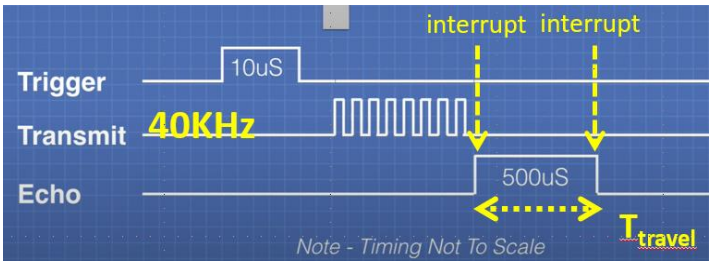
In the class project, you are going to design a self-driving toy vehicle. As shown below, the goal is self-driving the periphery (black line) of the track **5 times, as fast as possible**. The track will be provided during the project presentation in class. You should demo your work after presentation. You are allowed to use C and/or assembly.





Please use **interrupts** (instead of polling) wherever possible. You are free to use any I/O devices in our SoC, except the following minimum requirements;

- **Ultrasonic Sensor (**and/or **LiDAR)**: measure the distance to the obstacle in front
  - Use **interrupt** to detect the transitions of the echo input from the Ultrasonic sensor
  - You don't have to use interrupt for the LiDAR sensor



- **Accelerometer**: measure the car speed **every 200ms**
  - Use **interrupt** to read data from I2C
  - The measured distance should be sent to PC via **Bluetooth module**, to draw the speed graph after 5 turns

- **Line tracking sensor**:
  https://arduinomodules.info/ky-033-line-tracking-sensor-module/
  - Use the sensor to avoid the deviation from the driving path
  - Use **interrupt** to detect the deviation from the driving path

- **Timers**:
  - Use Timers to generate PWM (Pulse Width Modulation) signals for the motor control

What and How to submit:

1. Upload the zipped everything (Microchip studio project & PowerPoint) to Blackboard.
   - PowerPoint slides should have detailed description of H/W and S/W design.

2. Upload video clip (10-min?) to YouTube and provide the link to Blackboard. Your video clip should have **at least** the following contents:
   - Your smiling face
   - Understandable explanation of HW and SW
   - Demo

Note: This is an individual project. You are welcome to discuss, but DO NOT COPY solutions. **The plagiarism will not be tolerated!**

- Task Control Block (TCB) or Process Control Block (PCB)

When a task is created, it is assigned a task Control Block (TCB). TCB is a data structure that is used by OS to maintain the state of a task when it is preempted. When the task regains control of the CPU, the TCB allows the task to resume execution exactly where it left off. The TCB is initialized when a task is created. An example TCB structure used in µC/OS-II (a Real-Time Operating System) is shown below

```
struct os_tcb {

  OS_STK       * OSTCBStkPtr ; // a pointer to the current top-of-stack for the task
  struct os_tcb * OSTCBNext; // for doubly linked list of TCBs
  struct os_tcb * OSTCBPrev; // for doubly linked list of TCBs
  INT16U       OSTCBDly;     // a task needs to be delayed for this amount of clock ticks
  INT8U        OSTCBStat;    // task state: Dormant, Ready, Running, Waiting
  INT8U        OSTCBPrio:    // task priority

} OS_TCB
```

In this class project, things are much simpler because we have only 3 tasks, all tasks are always ready to run, and the priorities of the tasks are the same. Thus, it would be enough to save the following registers to TCB when context-switched.

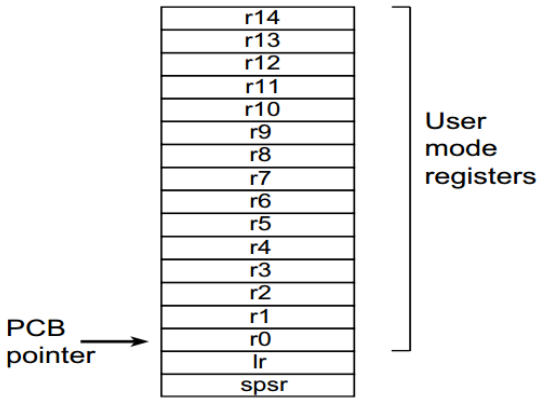


Figure 6-2 PCB layout

- One thing you have to be careful: The fact that User mode registers (because the tasks are running in User mode) have to be stored in PCB means that the ISR (or scheduler) in the IRQ mode should have access to the registers in the User mode. To make your life easier, ARM provides optional suffix (^) for LDM and STM. With that suffix, the User mode registers (instead of the current (IRQ, FIQ…) mode registers) are accessible in other (IRQ, FIQ…) mode.

For example, assume that the following STM instruction is executed in ISR (Interrupt Service Routine), which is running in IRQ mode.

STM sp, {r0-lr}^ // store r0 ~ r14 **User-mode registers** to stack.