CprE 288 – Introduction to Embedded Systems
(Lab 2 Overview)

Instructors:
Ian McInerney
Open Interface

- Program is on the MCU (ATmega128 processor)
- Motors for movement are on the iRobot
- Communication occurs over a standard RS232 serial port using UART0
- This communication has been abstracted by using the open interface
Open Interface

- Open Interface makes it so you don’t have to “see” the serial communication
- You simply call functions that handle the serial part for you

http://class.ece.iastate.edu/cpre288
// Allocate a sensor struct
oi_t* oi_alloc();

// Initialize the serial communication
void oi_init(oi_t *self);

// Update the oi_t sensor struct
void oi_update(oi_t *self);

// Set velocity of each wheel in mm/s (value should be between -500 and +500)
void oi_set_wheels(int16_t right_wheel, int16_t left_wheel);
Open Interface

• Initializing the serial connection

// Make sure the iRobot is **powered on**

```c
oi_t* sensor_status = oi_alloc();  // allocate memory
oi_init(sensor_status);           // initialize
```

http://class.ece.iastate.edu/cpre288
Open Interface

- `oi_t* sensor_status`
  - It's a struct for keeping the state of the iRobot.
  - Necessary since the status of sensors can only be current if serial communication is used.
  - Call `oi_update(sensor_status)` to refresh the members of the struct.

```c
typedef struct {
    // Boolean value for the right bumper
    uint8_t bumper_right;
    // Boolean value for the left bumper
    uint8_t bumper_left;
    // Boolean value for the right wheel
    uint8_t wheeldrop_right;
    // Boolean value for the left wheel
    uint8_t wheeldrop_left;

    // ... a lot more variables
} oi_t;
```
Move the Robot Forward

```c
#include "open_interface.h"
#include "util.h"

void main() {
    oi_t *robot = oi_alloc();
    oi_init(robot);

    ...  // call a function to move robots

    free(robot);
}
```
```c
#include "open_interface.h"
#include "util.h"

int move_forward(oi_t *self, unsigned int distance_mm)
{
    oi_set_wheels(..., ...); // set the speed of both wheels
    int sum = 0;
    while (sum < distance_mm) {
        oi_update(self);
        sum += self->distance;
        // optional check for bump sensors
    }
    oi_set_wheels(..., ...); // stop the robot

    return sum;
}
```
Lab 2, Part II. Robots moving in a square
Lab 2, Part III. Bump detection
What you will learn:

– How to program robot behavior using a set of API functions
– How API functions simplifies a programmer’s job

Common approaches when working with I/O devices