Today's Lecture

- Announcements/reminders
- HappyBoard
- JoyOS
- Setup and example

Check-Off for Assignment 1 by Midnight Tonight

- Check-off with any lab staff member
  - Know your TA/Organizer
  - Bring ID # if taking for credit
  - Bring car!

Assignment 2

- Will be posted later today
- Due Monday night at midnight
- Two tasks to complete:
  - Assemble and test Happyboard
  - Writing component
- Assignment 2 guideline will be available online
- Go to Workshop 1 to help complete the assignment
Other General Announcements

- Reminder: 6th floor doors to lab are locked after 5pm
  - To get to 6th floor lab, enter 5th floor lab and go up inside stairs
- Tell us if you want credit soon
- Check the website religiously for updates

HappyBoard Overview

- Atmega128 AVR
  - 4K RAM, 128K Flash
- 6 motor + 6 servo ports
- 8 digital I/O ports
- 16 analog input ports
- 4 encoders ports
- LCD, buttons, frob knob

More under technical specifications on course website!

Board History

- Handy Board
  - Created in 1995 by Fred Martin
- Happyboard
  - Created in 2006 by Ross Glashan
  - Modeled after Handy Board

General Idea
JoyOS

- Operating System for Happyboard
- Your code runs within JoyOS
- Gateway to the Happyboard

What JoyOS does for you

- Manages multiple threads
  - More details in lecture #3
- Communicates directly with HappyBoard hardware
  - FPGA, SPI, AVR ports, ADC, etc...
  - Provides simple API to talk w/ board
    - analog_read(), motor_set_vel(), etc...
- Libraries

More JoyOS Info

- Written in C using the avr-libc library
- Preemptive multitasking
- No MMU!
  - Can't protect memory from corruption

General Idea

Your code!
Actuator functions

// set "motor" to "vel"
void motor_set_vel (uint8_t motor, uint16_t vel);

// set "motor" to break mode
void motor_break (uint8_t motor);

// return current drawn by "motor"
uint16_t motor_get_current_MA (uint8_t motor);

// set "servo" position to "pos"
void servo_set_pos (uint8_t servo, uint16_t pos);

Sensor functions

// read digital sensor on input "port"
uint16_t digital_read (uint8_t port);

// read the analog sensor on input "port"
uint16_t analog_read (uint8_t port);

// read the encoder count on "encoder"
uint16_t encoder_read (uint8_t encoder);

// reset the encoder count for "encoder"
void encoder_reset (uint8_t encoder);

// read measured theta from gyro driver
double gyro_get_degrees ();

Getting ready to develop

- Need development environment
- Tools
  - Available for MacOS X, Windows, Linux
  - Used to compile code and program board
  - More details in “HappyBoard Manual”
  - We’ll setup lab computers

Loading code

- Procedure:
  - Compile code on workstation
    - Make sure there are no Errors
  - Turn on board while holding STOP
    - Should say “Happyboot v0.1”
  - Upload with programmer
    - Write each instruction to the processor
  - Restart board
Board startup info

- Power ON
- HappyBoot
- JoyOS main()
- Prepare for match start
- Your code
  - Calibration usetup()
  - Match code umain()

umain()

- The user main()
- Has it's own thread, can create new threads
- Houses autonomous code
- When returns the board halts
  - No reason to ever do this---we'll provide a good way to halt after 60 seconds.

Sense-Plan-Act Paradigm

- Match code umain()
- Sense
- Plan
- Pause
- Act

Wait... why am I pausing?

- Pause gives more CPU time to other threads
- Not necessary, but good practice
- Speed of CPU >> Mechanical response of robot
Example code structure

```c
int umain () { // start of match
    while (1) { // loop forever
        sense (); // read sensors
        plan (); // figure out what to do
        act (); // drive motors
        pause (10); // pause 10 milliseconds
    }
    return 0;
}
```

Methods to Drive Straight

- Shaft encoding
  - Relies on initial alignment
  - Relatively fast
  - Can be tricked by slipping
- Line following
  - Robust
  - Relatively slow
- Wall following
  - Requires continuous stretch of wall
  - Can be fast
- Gyroscope
  - Very fast
  - Instantaneously accurate
  - Can drift over time

Let's use the gyro!

```c
int umain () { // start of match
    int left_speed, right_speed;
    while (1) { // loop forever
        double deg = gyro_get_degrees();
        if (deg < 0) {
            left_speed = 140;
            right_speed = 160;
        } else {
            left_speed = 160;
            right_speed = 140;
        }
        motor_set_vel (0, left_speed);
        motor_set_vel (1, right_speed);
        pause (10);
    }
    return 0;
}
```

Simple program
Simple program

int main () {  // start of match
  int left_speed, right_speed;
  while (1) {  // loop forever
    double deg = gyro_get_degrees();
    if (deg < 0) {
      left_speed = 140;
      right_speed = 160;
    } else {
      left_speed = 160;
      right_speed = 140;
    }
    motor_set_vel (0, left_speed);
    motor_set_vel (1, right_speed);
    pause (10);
  }
  return 0;
}

- Isn't great...
  - Only drives towards 0 degrees
  - zig-zags
- Better ways to do this
  - PID Feedback
    - Learn more on Friday!

Purchasing Tools

- A number of staff tools necessary for electronics assembly will be available in lab
  - Everyone shares these tools, don’t hoard them!
- A limited number of tools is available for purchase to use outside of lab
  - Helping Hands $8
  - Wire Cutters $3
  - Soldering Iron $4
  - Needle Nose Pliers $4
  - Diagonal Cutters $5
- Tools Package $20

What's Next

- Workshops begin tomorrow
  - Offered at 1pm and 7pm each day
  - Sign-up sheets posted in lab
  - Workshop 1 this week, rest of the workshops next week
- Workshop 1 – Thursday (Tomorrow) and Friday
  - Takes place in 38-600 (our lab)
  - Happyboard Assembly - bring soldering supplies if you have them
  - Come to the workshop to receive your Happyboard and batteries
- Workshop 2 starts on Monday
- Workshop 3 starts on Tuesday

EOL

Questions?