Assignment 1: General Comments
- Sensor store and $30 Electronics Rule
- Electronics review
- Handy Board hardware and interface
- Sensors and motors
- Interactive C development environment
- Assignment 2 handed out

LECTURE 2: Building the Basic Robot
Teams $\textbf{NOT}$ Checked Off

- You know who you are!
- We think you are:
  - 6, 47, 48
Updates

• Course notes
• Assignment 2
Rules Clarifications

• Scoring moving balls
  – We score the match once all game objects have stopped moving
• Programming the HandyBoard without IC
  – You can try (but you shouldn’t)
• Position and orientation?
  – You can’t tell your robot about position/orientation
• How many servos?
  – We’ll talk about that in a second
• Can we melt rubber bands and dip LEGO in them?
  – No.
• Can we add things like a PIC?
  – Yes.
Other general announcements

• Don’t wander into the office
  – If you need something, ask a staff member (should be nearby)

• Staff parts (motors, sensors, etc...)
  – Please don’t take these
  – They’re for looking at or use where they live
The Contest: Points and Dollars

- Two sensor budgets for every team
  - 30 (of your) dollars
  - 20 “sensor points”

- See Chapter 2 of course notes for more details
Spending Your Money ($30 Electronics Rule)

- May be spent on electronics and actuators (and related parts)
  - Sorry, you can’t buy more LEGO
  - We sell things at cost – our prices are probably the best you’ll find
  - Provide receipts, or fair estimates; and make sure we agree before you buy
  - General rule of thumb – we should be able to get 500 of your part at the same price you’re claiming
  - If you’re getting free samples, they don’t count as free
- Make sure you know what you are doing
- Check out our stuff before you start looking elsewhere
- Tools do not count towards the allotment
- Replacements for servos do not count towards your $30 allotment—only based on what’s on your final robot
A Little More About the Servos...

- Yes, you have 6 servo ports
- But you can only have 4 (or maybe 5) servos
- Each cost $10
  - Except the old ones, which are $7
- To replace broken, $15; does not count towards $30 allotment
- For those wishing to get the servo elsewhere:
  - Futaba S3003
Sensor Points

- Sensor points can go to mechanical switches, sensors, LEDs, and motors
  - We have some sensors you haven’t seen yet
  - Full catalog of components available are displayed by the 6.270 office
- Generally, LEDs, mechanical switches, and phototransistors cost 1 point
- Motors cost 1 point
- Distance sensors costs 3 points
- Be sure to test similar sensors; not all are the same
Trading

- You can trade sensors using our point scheme
- You can trade sensors in your original kit with the sensor store, but we’ll be picky (“like-new condition”)
- If you want to purchase a replacement from us, it’s $5 or the cost of the part, whichever is higher (won’t cost towards the $30; we’re not an electronics store)
- **NO LEGO TRADING**, unless it’s for color
  - ... well, don’t trade gray for black pegs, either
What Are Sensors?

- Devices that change resistance due to
  - Light
  - Pressure
  - Position (angle)
  - etc...

- Range is limited (with IC: analog 0 to 255; digital 0 to 1)
HandyBoard Sensor Inputs

\[ V = 5 \, V \]

\[ R = 47 \, k\Omega \]

IN to ADC
HandyBoard Sensor Inputs

• HandyBoard has digital and analog hardware ports
  – You can read both types of ports as digital or analog values
  – BUT, digital ports read as analog values always give 0 or 255
  – Analog ports read as digital values use a cutoff to decide if it is 0 or 1
  – Analog ports: 0-6, 16-31
  – Digital ports: 7-15
Expansion Board Sensor Inputs

- Expansion inputs are multiplexed
- Need to read inputs twice when using a different expansion port

\[ V = 5 \text{ V} \]
\[ R = 47 \text{ k}\Omega \]

\[ IN_0 \quad \text{to ADC} \]
Sensors

- Potentiometer
- Phototransistor
- Breakbeam
- Distance
Sensor Connection

VCC  SIGNAL  GND
Sensors

- Bumpers
- Internal sensors
- One terminal to GND, other to Signal (IN)
- NC (normally closed) vs. NO (normally open)
Potentiometers (Variable Resistors)

- Useful for precise shaft encoding
Phototransistor

- More sensitive to IR than to visible light
- Polarized
- More in Lecture 3
Breakbeam Sensors

• Shaft Encoding
• Works on certain HB ports
• Count number of interruptions
• More in Lecture 3
Sharp Distance Sensor

- Useful for measuring distances from 6.5” to a couple feet
- 3 sensor points
- More in lecture 3
Sharp Distance Sensor

- Response curve

![Response curve graph](image)
The Gyroscope!

- Now, a word from Analog Devices
Motors

- Two kinds of motors available
- Need to “LEGOize” motors
- Can use glue or tape to mount them
Motor Connections

- Three pins
- Use first and third pin
- Current can flow either direction, motor runs in either direction
Interactive C

- Originally developed for 6.270 by Randy Sargent
- Supports Handy Board, RugWarrior and RugWarrior Pro
Where to Develop

- Work at Lab
  - IC 3.1
  - Laptops in lab

- Work at Home
  - IC 3.2 for Windows
    - http://www.kipr.org
Starting Off: Download pcode

- Initially do this to load machine instructions and libraries to HB
  - IC provides some libraries, 6.270 provides more
  - Be sure HB has been charged
- May want to do this if HB is acting strangely
- Set HB to download mode (hold STOP and turn on)
Starting IC

- Add 6.270 locker
- Type: ic
- Make sure HB is on
  - LCD: IC v3.1
    SMOOTHPWM with beating heart
- Prompt: c>
Basic Commands

• `load <filename>`
  – IC looks in current directory first, then IC library path
  – Several files may be loaded into IC at once
  – A program can be defined in multiple files
  – Downloads all loaded files to HandyBoard

• `unload <filename>`
  – Unloads the named file
  – Re-downloads remaining files
Basic Commands

• `help`
  – Help screen of IC commands
• `quit`
  – Exits IC
  – `<Ctrl-C>` also works
Editing

• IC has a line editor and command history
• Scan through history with ↑ and ↓ arrows or <Ctrl-P> (previous command) and <Ctrl-N> (next command) respectively

**KEY MAPPINGS**

- <Ctrl-A> beginning of line
- <Ctrl-E> end of line
- <Ctrl-B> back one character
- <Ctrl-F> forward one character
- <Ctrl-D> delete character
- <Ctrl-K> kill line
• At prompt, C-language expression may be entered
• End with semicolon
  – Example:
    C> 6 * 10;
    Downloading 7 bytes (addresses C200–C206): 7 loaded
    Returned <int> 60

• Can also evaluate series of expressions by creating a block with curly braces
  – Example:
    C> { int i = 10; printf( "%d\n", i ); }
Program Development

- Develop in favorite editor (vi, emacs, Notepad)
- In IC: `load <filename>`
- Reboot HB by turning it off then on
How the HB Runs Code

• When turned on, HB runs `main()` function
• Reset HB without running `main()`
  – Hold START button while turning on
  – HB will display IC 3.1 SMOOTHPWM
  – Files are not lost
How the HB Runs Code

• Global variables initialized whenever reset condition occurs
• Reset occurs when
  – New code downloaded
  – `main()` is run
  – System hardware is reset
• Local variables initialized when function containing them is called
Conflicting Files

- Occurs when multiple files downloaded have a `main()` (or any other) function
- Error occurs
- See the manual for more details
Persistent Global Variables

- Special uninitialized global variable
- Preface with keyword `persistent`:
  ```
persistent int i
  ```
- Initial values cannot be specified in declaration
- These variables keep their state when:
  - HB is turned off and on
  - `main()` is run
  - System reset occurs
Persistent Variables

- If declared at the beginning of the code, before any function or other non-persistent globals, they will be re-assigned to the same location in memory when code is re-compiled.
- Can preserve values over multiple downloads.
- If program is divided into multiple files, all persistent variables should be in one file.
  - File should be placed first in the load ordering of the files.
Why Use Persistent Variables?

- Store calibration and configuration values that do not need to be re-calculated every time
- Robot learning algorithms
List Files (.lis)

- List multiple files needed for download in a .lis file
- Remember: if you have persistent variables in code, ensure that file containing them is listed first
  - `load <listfile>` loads files in the order prescribed
Printing to LCD

• Only 31 characters \((16 \times 2 - \heartsuit)\)
• Characters printed beyond final character position are cut off
  • `printf()` treats the LCD screen as one long line instead of two
• Cannot print long 32-bit integers
Arrays and Pointers

- Arrays are one-dimensional only
- Pointers to only data items and arrays
Motors

- `void fd(int m)`
- `void bk(int m)`
- `void off(int m)`
  
  motor ports: $0 \leq m \leq 5$

- `void alloff()`
- `void ao()`

- `void motor(int m, int speed)`
  
  $-100 \leq speed \leq 100$  
  
  100: full forward  
  
  -100: full backward
More About Motors

- \texttt{motor(m,0)} and \texttt{off(m)} are the same command
- Motor ports 4 and 5 do not have speed controls
  - They turn on full power
  - \texttt{speed} parameter does not matter
Servos

- `void disable_servos()`
- `void enable_servos()`
  - Must be called for servos to work
  - Disable servos when processor performance is needed
- `void servo(int port, int period)`
  - $0 \leq port \leq 5$
  - $0 \leq period \leq 4000 \ (\approx 180^\circ)$
Sensors

- **int analog(int port)**
  - Returns integer between 0 and 255
- **int digital(int port)**
  - Returns 0 (false) or 1 (true)
    - $0 \leq port \leq 31$
- You can plug analog sensors into digital ports and vice versa
Other Doodads

• `int stop_button()`
• `int start_button()`
  - Returns 1 (pressed) or 0 (released)

• `void start_press()`
• `void stop_press()`
  - Waits for button to be pressed and released
  - Beeps afterwards

• `int knob()`
  - Returns position of knob as integer between 0 to 255
Time

- `void reset_system_time()`
  - Reset time to 0 milliseconds
- `long mseconds()`
- `float seconds()`
  - 1 millisecond resolution
  - `int` vs. `float` (the period)
- `void sleep(float sec)`
  - At `sec` or a little longer than `sec` seconds
- `void msleep(long msec)`
  - At `msec` or longer than `msec` milliseconds
Tones

- `void beep()`

- And for the bored:
  - `void tone(float freq, float length)`
    - `freq` Hertz for `length` seconds
  - `void set_beeper_pitch(float freq)`
Assignment 2

- Due Thursday night (TOMORROW!) at 11:45 pm
- Four tasks to complete:
  1. Build and test the expansion board
  2. Build and test the RF receiver
  3. Build a robot (guidelines are enumerated in assignment)
  4. Program the robot to move around
- Pick up assignment after lecture
- Assignment 3 is dependent on the robot you made for Assignment 2
- The robot just needs to work: remember that it doesn’t have to be the best robot you’ve ever made
What’s Next

• Wednesday, January 5, and Thursday, January 6
• Workshop 3 – Electronics Assembly
  – How to solder
  – Soldering RF receiver (Assignment 2)
• Workshop 4 – Code & Sensors I: Basic Control and Robot Skills
  – Programming the HB (Assignment 2)
• Don’t forget to sign up on the 6th floor lab