Sensors, Designing Sensor Suites, and Sensor Fusion

COMP 4510
Fall 2019
Prof. Yanco
Sensors

• Measure internal or external state of the robot
• Internal state, also called proprioception
  – Battery level
  – Wheel rotations
  – Position of gripper (e.g., open, closed)
  – Etc.
• External state:
  – Location of objects with respect to the robot
  – Color of objects
  – Location of wireless base stations
  – Etc.
Different Types of Sensors

- Thermistor (Temperature Sensor)
- IR Sensor (Transmissive Type)
- IR Sensor (Reflective Type)
- Ultrasonic Sensor
- Gyroscope Sensor
- Accelerometer Sensor
- Rain Sensor
- Soil Moisture Sensor
- Phototransistor (Light Sensor)
- Water Flow Sensor
- Heartbeat Sensor
- Alcohol Sensor
- Color Sensor
- PIR Sensor
- Gas Sensor
- Smoke Sensor
- LM35 (Temperature Sensor)
- IR Receiver
- LDR (Light Sensor)
- Humidity Sensor
- Flex Sensor
- Touch Sensor
- Solar Cell Light Sensor
- Metal Detector
- Real Time Clock Sensor
- Vibration Sensor

Sensors

• Active
  – Emit energy into the environment to make observations
  – E.g., infrared, sonars
  – Don’t confuse active sensors (devices that emit energy) with active sensing (e.g., pan-tilt camera)

• Passive
  – Receive already existing energy from environment
  – E.g., bump sensor, video camera (without additional lighting)
Sensors

- Sensors which measure same form of energy and process it in similar ways form a modality
- Modalities refer to the raw input
  - Sound
  - Pressure
  - Temperature
  - Light
Types of Data Returned

- Range
  (e.g., distance to objects)
- Absolute position
  (e.g., latitude and longitude)
- Environmental
  (e.g., temperature or color of objects)
- Inertial
  (e.g., acceleration)
Sensors: Inertial Sensors

• Accelerometers
  – Measure acceleration in a single direction

• Gyroscopes
  – Measure change in orientation

• Compasses
  – Measure position relative to magnetic north

• Inclinometers
  – Measure orientation of the gravity vector
Sensors: Global Positioning

- Global Positioning Sensors (GPS) use satellite readings to determine location
- Differential GPS (DGPS) uses a base station to eliminate the intentional error in the satellite signals (not used much these days)
- Only works when you can see the satellites (i.e., not indoors)
Ranging Sensors: Sonar

• Sends an acoustic pulse and listens for echo
• Determines how far away an obstacle is based upon time of flight and speed of sound
• Can have problems with specular reflections
Ranging Sensors: Infrared

• Fast and inexpensive proximity sensors
• Emit an infrared pulse and detect the reflected signal
• Infrared can be washed out by bright light; unusable outdoors in most cases
Ranging Sensors: Laser Ranging

• Uses one of the following methods
  – Triangulation (relationship between outgoing light beam, incoming ray and its position on the film plane)
  – Time-of-flight (similar to sonar or infrared)
  – Phase-based (based on difference between phase of emitted and reflected signals)

• Low-power ranges a few meters
• High-power can range over a kilometer
• Many robot systems now using Kinect instead
Sensors: Cameras

• Images from cameras must be processed in order for the robot to use the information
  – Edge detection
  – Tracking movement
  – Finding particular colors
  – Distance of objects from the camera
RGB Color Space

- Lighting impacts color values!
HSV Color Space

- **Hue**, the color type (such as red, blue, or yellow);
  - Measured in values of 0-360 by the central tendency of its wavelength
- **Saturation**, the 'intensity' of the color (or how much grayness is present),
  - Measured in values of 0-100% by the amplitude of the wavelength
- **Value**, the brightness of the color.
  - Measured in values of 0-100% by the spread of the wavelength
Looking for Colors

• Can train on colors in a region of the image, then track that color

• Best to track colors in HSV color space (RGB is too lighting dependent)
Image Processing Pipeline

• Grab image
• Filter to smooth image
• Process for some property
  – Intensity changes for edges
  – Blobbing to find an area of a particular color
• Act on the results
Filtering Methods

- Median
- Mean
- Gaussian
Gaussian filter

- This filter $H$ is a good approximation to
- Properties of Gaussian
  - more weight to the center
  - good model of blurring in optical systems
  - $\sigma$ corresponds to width of the Gaussian
Gaussian Filter
Mean Blur

- Blurs the image by changing the color of the pixel being looked at to the mean value of the pixels surrounding it. The number of surrounding pixels being looked at is defined by the kernel parameter. If kernel is 3, then the pixel being looked at is the center of a 3x3 box, shown in the diagram.

![Diagram showing mean blur](image)
Mean Blur
Median Blur

- Blurs the image by changing the color of the pixel being looked at to the median value of the pixels surrounding it. The number of surrounding pixels being looked at is defined by the kernel parameter. If kernel is 3, then the pixel being looked at is the center of a 3x3 box, shown in the diagram.
Median Blur
Edge Detection: Sobel

\[
|G| = \sqrt{G_x^2 + G_y^2}
\]
Edge Detection: Sobel
Edge Detection: Canny

1. Apply Gaussian filter
2. Sobel edge detection
3. Find direction of edges
4. Relate edge direction to direction that can be traced in an image
5. Nonmaximum suppression used to trace along the edge in the edge direction to suppress any pixel value that is not considered to be an edge
6. Hysteresis used to eliminate streaking (breaking up of an edge contour)
Edge Detection: Canny
Color Filtering
Color Blobbing
Stereo Vision

Finding a 3D point cloud from a pair of 2D stereo images via stereo matching and disparity calculations.

From Brigit Scroeder
Analog Sensors

- **Range of values:**
  0 – 4095
- **Ports:** 0 – 5
- **Function:** `analog(port #)`
- **Sensors:**
  - Light
  - Small reflectance
  - Large reflectance
  - Slide sensor

Digital Sensors

- **Range of values:**
  0 (not pressed) or 1 (pressed)
- **Ports:** 0 – 9
- **Function:** `digital(port #)`
- **Sensors:**
  - Large touch
  - Small touch
  - Lever touch
KIPR Robotics Controller sensor ports

Digital Sensors
Ports # 0 - 9

Analog Sensors
Ports # 0-5
There are many digital sensors in your kit that can detect touch...

Select the one that can be easily attached *and* can easily detect the objects.

- Large touch
- Small touch
- Lever touch
Plug in a Touch Sensor

Plug your touch sensor into digital port 0
Scroll down to the digital sensor and read the value when your touch sensor is pressed and when it is not pressed.
Use the sensor graph

![Diagram showing the sensor graph interface]

- Click on the 'Motors and Sensors' tab.
- Select 'Sensor Graph' to view sensor data.
Remember Your Sensor Functions

You call for the analog sensor value with a function

- You have 6 analog ports (0-5)

\[ \text{analog(Port#)} \quad \text{analog(1)} \]

You call for the digital sensor value with a function

- You have 10 digital ports (0-9)

\[ \text{digital(Port#)} \quad \text{digital(8)} \]

**NOTE:** when you call these functions they return an INTEGER value into the “code” where they were called at the time the code is run.
• Returns the analog value of the port (a value in the range 0-4095). Analog ports are numbered 0-5.

• Light sensors, slide, range finders and reflectance are examples of sensors you would use in analog ports.
This sensor is really a short range reflectance sensor. There is an infrared (IR) emitter and an IR collector in this sensor. The IR emitter sends out IR light and the IR collector measures how much is reflected back.

Amount of IR reflected back depends on surface texture, color and distance to surface

This sensor is excellent for line following

Black materials typically absorb IR and reflect very little IR and white materials typically absorb little IR and reflect most of it back

- **If this sensor is mounted at a fixed height above a surface**, it is easy to distinguish a black surface from a white surface
- Connect to analog port 0 through 5
IR Reflectance Sensor Behavior

Amount of reflected IR depends on surface texture, color, and distance to surface (higher values mean less IR indicating a dark surface or a drop off)
Problems with Sensors

• Noisy
• Return an incomplete description of the environment
• Can not be modeled completely

• We will come back to these issues throughout the term (and your robotics career), particularly when we talk about
Constructing a Sensor Suite

Consider the following for each sensor:

1. Speed of operation
2. Cost
3. Error rate
4. Robustness
5. Computational requirements
6. Power, weight and size requirements
Attributes of a Sensor Suite

Consider for entire suite:

1. Simplicity
2. Modularity
3. Redundancy
   - A sensor might be noisy or inaccurate by itself
   - False positives / False negatives
   - Complementary sensors can be used to provide disjoint information about a percept
Fusing Sensor Data: Sensor Fission
Fusing Sensor Data: Action-Oriented Sensor Fusion
DC Motors
Wallaby motor ports

Motor Labels are on the Case

Motor Ports 0, 1, 2, and 3
Plugging in motors

- **Motors** have red wire and a black wire with a **two-prong plug**.
- The Wallaby has 4 motor ports numbered **0 & 1** on left, and **2 & 3** on right.
- When a port is powered (receiving motor commands), it has a light that glows **green** for one direction and **red** for the other direction.
  - Plug orientation order determines motor direction.
  - By convention, **green** is **forward** (+) and **red** is **reverse** (−)
    - Unless you plug in the motors “backwards”.

---

Motor Port #2  
Motor Port #3  
Drive motors have a two-prong plug.
Plugged in motors

DemoBot Motor Ports 0 (right wheel) and 2 (left wheel)
You want your motors going in the same direction; otherwise, your robot will go in circles!

- **Motors** have a red wire and a black wire with a **two-prong plug**.
- There is no left side or right side.
- You can plug these in two different ways:
  - One direction is clockwise, and the other direction is counterclockwise.
  - The red and black wires help determine motor direction.
There is an easy way to check this!

• Manually rotate the tire, and you will see an LED light up by the motor port (the port # is labeled on the board).
  • If the LED is green, it is going forward (+).
  • If the LED is red, it is going reverse (−).

• Use this trick to check the port #’s and direction of your motors.
  • If one is red and the other is green, turn one motor plug 180° and plug it back in.
  • The lights should both be green if the robot is moving forward.
Use the Motor Widget
There are several functions for motors. We will begin with `motor`.

```plaintext
motor(0, 100);
// Turns on motor port #0 at 100% power.
// Select any power between -100% and 100%.

msleep(# milliseconds);
// Wait for the specified amount of time.

ao();
// Turn off all of the motors.
```

**Motor port #**

(between 0 and 3)

A **positive number** should drive the motor **forward**; if not, rotate the motor plug 180°.

A **negative number** should drive the motor **reverse**.

If two drive motors are plugged in in opposite directions from each other, then the robot will go in a circle.
Using `motor` and `ao`

```c
int main()
{
    motor(0, 100);
    motor(2, 100);
    msleep(2500);
    ao();
    return 0;
}
```
Each motor used by the DemoBot has a built-in motor position counter, which you can use to calculate the distance traveled by the robot!

```c
get_motor_position_counter(0) — OR — gmpc(0)
// Tells us the number of ticks the motor on port #0 has rotated.
// Note: “gmpc” is shorthand for “get_motor_position_counter”.

clear_motor_position_counter(0); — OR — cmpc(0);
// Resets the tick counter to 0 for the motor on port #0.
// Note: “cmpc” is shorthand for “clear_motor_position_counter”.
```

- The motor position is measured in “ticks”.
- Botball motors have approximately 1400 ticks per revolution.
- Use wheel circumference divided by 1400 to calculate distance!
Seeing Counters on Wallaby

You can access the Motors from the Motors and Sensors section

• This is very helpful to test your motors and see the actual motor position counters "in action"
Select motor port (allows you to select the motor of your choice)

To clear (reset) the counter

Motor Position in “ticks”

Use your hand to rotate the robot’s wheel (plugged into port 0) and watch the position counter.

What happens if you turn the wheel in the opposite direction?

You can also place your robot on a surface and roll it forward to measure the # ticks from a starting position to another location or object.
Using motor position counter functions

```c
int main()
{
    clear_motor_position_counter(2);
    while (get_motor_position_counter(2) < 1400)
    {
        motor(0, 50);
        motor(2, 50);
    }
    ao();
    return 0;
}
```

How many revolutions will the motor rotate?
Until you are familiar with the functions that you will be using, use this **cheat/hint sheet** as an easy reference.

Copying and pasting your own code is also very helpful.

```c
printf("text\n");
// Prints the specified text to the screen
msleep(# milliseconds);
// Another name for wait_for_milliseconds (identical)
omotor(port #, % velocity);
// Turns on motor with port # at specified % velocity
motor_power(port #, % power);
// Turns on motor with specified port # at specified % power
mav(port #, velocity);
// Move motor at specified velocity (# ticks per second)
mrp(port #, velocity, position);
// Move motor to specified relative position (in # ticks)
ao();
// All off; turns all motor ports off
enable_servos();
// Turns on servo ports
disable_servos();
// Turns off servo ports
set_servo_position(port #, position);
// Moves servo in specified port # to specified position
wait_for_light(port #);
// Waits for light in specified port # before next line
wait_for_touch(port #);
// Waits for touch in specified port # before next line
analog(port #)
// Get a sensor reading from a specified analog port #
digital(port #)
// Get a sensor reading from a specified digital port #
shut_down_in(time in seconds);
// Shuts down all motors after specified # of seconds
```
int main()
{
    printf("Hello ");
    msleep(2500); // wait for 2500 ms
    printf("what is your name?\n");
    return 0;
}

What is this?

Another name for \texttt{wait\_for\_milliseconds()} is \texttt{msleep()}. It is identical and shorter to type, but more difficult to remember.

\texttt{msleep(2500)} is the same as \texttt{wait\_for\_milliseconds(2500)}. 
Access the Wallaby documentation by selecting the *Help* button in the KISS IDE.
Robot driving hints

Remember your # line:
positive numbers (+) go forward and negative numbers (−) go in reverse.

Driving straight: it is surprisingly difficult to drive in a straight line...
- **Problem:** Motors are not exactly the same.
- **Problem:** The tires might not be aligned perfectly.
- **Problem:** One tire has more resistance.
- **Solution:** You can adjust this by slowing down or speeding up the motors.

Making turns:
- **Solution:** Have one wheel go faster or slower than the other.
- **Solution:** Have one wheel move while the other one is stopped.
- **Solution:** Have one wheel move forward and the other wheel move in reverse (friction is less of a factor when both wheels are moving).
Next Tuesday: Meet in DAN 407

• Robot kit distribution
• Short lab to test kit parts
Using the Wallaby
Charging the Controller’s Battery

• For charging the controller’s battery, **use only the power supply which came with your controller.**
  • It is possible to damage the battery by using the wrong charger or excessive discharge!

• The standard power pack is a **lithium iron (LiFe) battery**, a safer alternative to lithium polymer batteries. The safety rules applicable for recharging any battery still apply:
  • Do **NOT** leave the battery **unattended** while charging.
  • Charge in a cool, open area away from flammable materials.
All connections are as follows:

- **Yellow to Yellow** (battery to controller)
- **White small to White small** (charger to battery)
  - Yours may vary slightly, use caution unplugging
- **Black to Black** (motors, servos, sensors)
KRC Wallaby Controller Guide

KIPR Robotics Controller Wallaby

- Color Touch Screen
- 2 Servo Motor Ports (Port # 0 & 1)
- 2 Motor Ports (Port # 0 & 1)
- 10 Digital Sensor Ports (Port # 0 - 9)
- 2 Motor Ports (Port # 2 & 3)
- 2 Servo Motor Ports (Port # 0 - 5)
- 6 Analog Sensor Ports (Port # 0 - 5)
- Download port (micro USB)
- Power Switch
- USB
- Micro HDMI
- Power (external battery connection)
• The KIPR Robotics Controller – Wallaby, uses an external battery pack for power.
  • It will void your warranty to use a battery pack with the Wallaby that hasn’t been approved by KIPR.

• Make sure to follow the shutdown instruction on the next slide. Failure to do so will drain your battery to the point where it can no longer be charged. If you plug your battery into the charger and the blue lights continue to flash then you have probably drained your battery to the point where it cannot be charged again. You can purchase a replacement battery from www.botballstore.org.
• From the Wallaby Home Screen press *Shutdown*
  • Select *Yes*

• Go to your Wallaby screen and check to see if it is *halted* (If your Wallaby shows to be unable to be halted, rerun your last program either to completion or just start and stop it, and this should clear up any problem)

• Slide the power switch to off AND **unplug the battery**, using the yellow connectors, being careful not to pull on the wires
Connect the Wallaby to your computer at Workshop and Tournament

- **Connect the Wallaby** to your computer using **USB Cable**
  1. Plug battery into Wallaby - YELLOW TO YELLOW.
  2. Turn on the Wallaby with the **black switch on the side**

1. Once your Wallaby has booted, the Wallaby will appear in the list of available Ethernet connections for your computer.
2. If you get a message about the driver raise your hand for help or go to the team home base: *Troubleshooting-* > **USB driver** for instructions
1. Launch your web browser (such as Chrome or Firefox, but not Internet Explorer) and power up your Wallaby.

2. Copy this IP address into your browser’s address bar followed by “:” and port number 8888; e.g.,

   192.168.124.1:8888

   IP address     Port #

3. Note that USB cable IP address is 192.168.124.1:8888

4. The user interface for the package will now come up in your browser.

5. TEST THIS at the workshop

   a. See Team Homebase -> 2018 Resources -> Troubleshooting -> USB Driver
Connect the Wallaby to your computer, Smart Phone or Tablet At School

- Connect the **Wallaby** to your Browser device via Wi-Fi
- This is great at home or School
- **Not recommended at Large Workshops or any Tournament**

1. **Turn on the Wallaby with the black switch on the side**
   a. Note: the actual version number you see *most likely will be v23 (or higher)*

2. Use the info (Wallaby SSID # and Password), from the about page, to connect via Wi-Fi
Connection

When you are connected to your Wallaby, your device may give various errors; “no internet connection” or “connected with limited.”

In the **bottom right corner** of the KIPR IDE there is an icon that shows if you are still connected to the Wallaby.

![Connected](image1.png)  ![Not Connected](image2.png)
1. Launch a web browser such as Chrome or Firefox (Internet Explorer **will not work**) and power up your Wallaby. Connect to the Wallaby via Wi-Fi.

2. Copy this IP address into your browser’s address bar followed by “:” and port number 8888; e.g.,

   **192.168.125.1:8888**

   IP address    Port #

3. The user interface for the package will now come up in your browser.
   a. **Note:** during competitions use the USB cable connection (IP address: **192.168.124.1**)

4. You may use a computer, tablet or even a smart phone through Wi-Fi.
To make it easier for you to learn and use a programming language, KIPR provides a web-based Software Suite which will allow you to write and compile source code using the C programming language.

The development package will work with almost any web browser except Internet Explorer.
1. Add a new user folder by clicking the + sign in the Project Explorer.
2. Name your new user folder by the student’s name to help organization. All of your different projects will go into this user folder.

3. Click Create to complete.
1. Go back to **Project Explorer** and select the **User Name** you created from the drop down. This is the folder you created.

2. Click **+Add Project**. You are adding a project to your folder.
1. Give your project a **descriptive name**
   - **Note:** you will have a lot of student’s projects, so consider using their first name followed by the name of the activity.

2. Give a descriptive Source File Name as well. The Source File needs to end with a `.c`
   - Then press the **Create** button.

![Create New Project](image)
1. Click the **Compile** button for your project and, if successful, click **Run** so you can run your project to see if it works.

NOTE: When you compile, your project is automatically saved.
Starting another project

Note: one project = one program.

• Click the **Add Project** button or click the **Menu** button to return to the starting menu.

• Proceed as before.

• The **Project Explorer** panel will show you all of the user folder projects and actively edited files.