Project 2 Lessons
Project 2 Lessons

Functions can be abstractions
• Hide details from their “callers”
• In our case: we are hiding the details of the analog interface and how to interpret the analog values
• Functions should adhere to their specification and do no more
Hovercrafts

Have been delivered!

• Every group has two batteries. Use only one at a time
  • Working on new mounts for the batteries
  • Common battery chargers are being installed in the lab. Play nice

• New top plates have been installed
  • You will need to add Velcro tape to mount your circuit boards
Hovercrafts

Battery + switch on = system is energized

• 3 chassis LEDs turn on
• Thick red/black wires: +9V connection to the battery (keep these capped for now)
• Think red/black wire pair: +5V
  • We will connect to Vin soon, but have Jack convert your Teensy over to be able to use this power (else could damage your laptop)
  • After conversion: can only power Teensy from battery
Safety

• Only (un)plug batteries with the switch turned off
• Keep fingers off the lower deck
• Fuse should protect you from short circuits
  • We are using 10A fuses for now. Extras will be in the spare parts bin
• If you see smoke: inform one of us
  • And don’t keep popping new components into your circuit…
Project 3: Lateral Velocity Sensing
Project 3: Lateral Velocity Sensing

Each hovercraft has 3 downward-looking cameras at different angles

• Connect to a Serial Peripheral Interface (SPI)
  • High-speed serial bus

• When you query a camera, it will tell you how many pixels of “slip” have happened since the last time you asked
  • Both X and Y components

• With two or more cameras, we can estimate how far the craft has moved in three dimensions
Component 1: Physical Interface

Common across all cameras:
- Black: Ground
- Red: +5V Power
- Blue: MISO (Arduino pin 12)
- Orange: MOSI (Arduino pin 11)
- Green: SCL (Arduino pin 13)
- Gray: Reset (choose an unused digital pin)

Each camera has a yellow select line (choose a unique, unused digital pin)
Component 2: Interface Function

Implement the function:

```c
void accumulate_slip(int32_t adx[3], int32_t ady[3])
```

• Queries each of the cameras
• If there is slip, then it adds the latest slip to the accumulated slip
Component 3: Data Collection

• Record 10 repetitions of the accumulated values for three types of movement: forward 1m, leftward 1m, rotate clockwise 360 degrees
• Store in a table
Component 4: Sensor Model

• We are estimating the parameters of functions of the forms of:

\[ X = a_0 + a_1 \cdot \text{adx}_1 + a_2 \cdot \text{ady}_1 + a_3 \cdot \text{adx}_2 + a_4 \cdot \text{ady}_2 + a_5 \cdot \text{adx}_3 + a_6 \cdot \text{ady}_3 \]

• where \(a_0 \ldots a_6\) are the coefficients of our function, and \(\text{adx}_?/\text{ady}_?\) are the accumulated slip values
Sensor Model

Use “Multi-Regression” to compute the parameters
• Handle dX, dY and dtheta separately
• Use all 30 data points to fit each of the three parameter sets
Part 5: Implement the Model

Implement the function:

```c
void compute_chassis_motion(int32_t adx[3], int32_t ady[3],
                             float[3] motion);
```

• Translate adx and ady into hovercraft motion
Part 6: Testing

Loop():
• Accumulate slip values
• Occasionally compute and report motion

Take five more samples of each motion type
• Graphically report mean, standard deviation of each dimension
Hints

• Start this project early
• Keep things simple