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

Project 3: Lateral Velocity Sensing

Project 3: Lateral Velocity Sensing

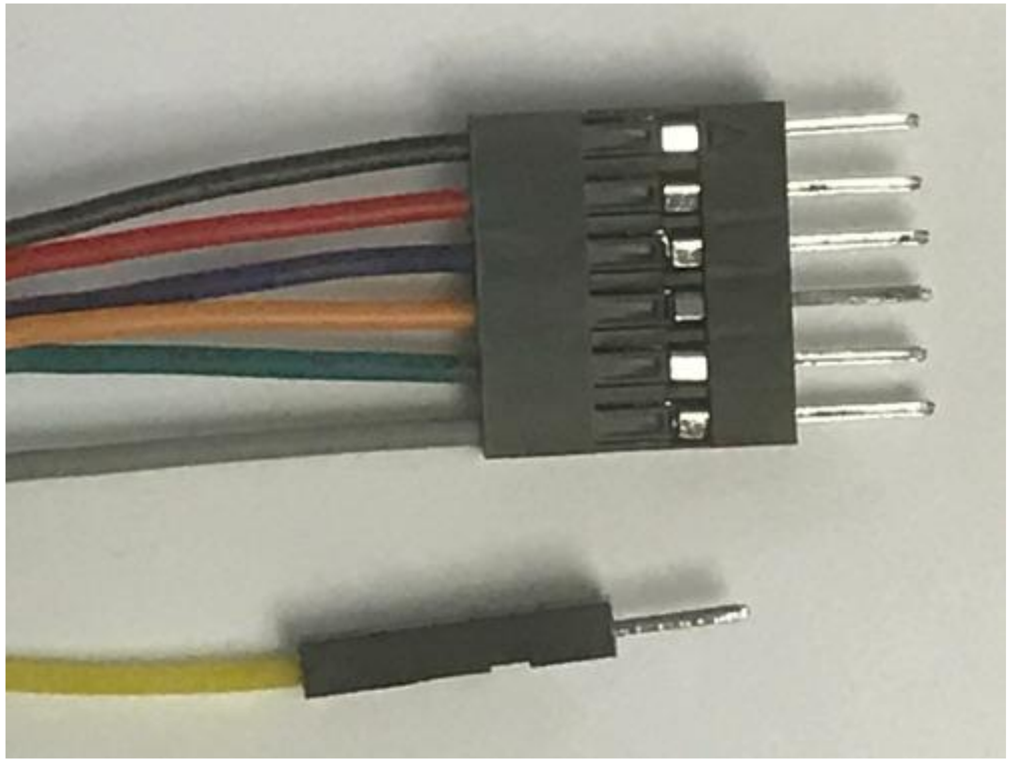
Each hovercraft has 3 downward-looking cameras at different angles

- 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: Supporting Types/Implementation

Top of program:

```
// Promise that we will implement this function later void void
void camera_step();

// Create a task that will be executed once per 50 ms
PeriodicAction camera_task(50, camera_step);
```

Loop:

```
void loop()
{
        // Check to see if it is time to execute camera_step()
        camera_task.step();
}
```

camera_step()

- This function is guaranteed to be called once per 50 ms
- For each call to this function:
  - Call accumulate_slip() to read the cameras and interpret the returned values

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

  - Once per second:
    - Print the accumulated slip values
  - If the character ‘c’ is received from the laptop, then set all the slip values to zero

Component 3: Interface Function

Implement the function:

```
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

Camera Interface

Top of program (example definitions):

```
// Global constants
// Total number of cameras
const int NUM_CAMERAS = 3;

// Select pins for the 3 cameras
const uint8_t CAMERA_SELECT[NUM_CAMERAS] = {8, 7, 10};

// Common reset pin
const uint8_t RESET_PIN = 9;

// Camera interface object
OpticalFlowCamera cameras(RESET_PIN);
```

accumulate_slip()

```
int8_t dx, dy;
uint8_t quality;
int result;

// For the ith camera:
result = cameras.readSlip(CAMERA_SELECT[i],
                                    dx, dy, quality);
```

New behavior in C++ (not seen in C):

- readSlip() will change the value of the variables dx, dy and quality

readSlip

```
result = cameras.readSlip(CAMERA_SELECT[i],
                                                    dx, dy, quality);
```

- If result == 0:
  - dx, dy and quality variables have been changed and can be used
- If result == -1:
  - readSlip() is not being called quickly enough
- If result == -2:
  - No slip has occurred; do not use dx, dy and quality

Component 4: 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: a total of 30 rows

Component 4: Sensor Model

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

```
dX = a1 * adx1 + a2 * ady1 + a3 * adx2
   + a4 * ady2 + a5 * adx3 + a6 * ady3
```

- where a1 ... a6 are the coefficients of our function, and adx?/ady? are the accumulated slip values

Sensor Model

Use “Multi-Regression” to compute the parameters

- Handle dX, dY and dtheta separately (one set of parameters for each)
- Use all 30 data points to fit each of the three parameter sets

Component 5: Implement the Model

Implement the function:

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

Translate adx and ady into hovercraft motion

- Inputs: adx, ady
- Output: motion

Component 6: Testing

camera_step() changes:

- Once per second: compute and print motion

Take five more samples of each motion type: move forward 1m, move left 1m and turn 360 degrees

- For each of dX, dY and dtheta (separately): plot mean and variance for each motion type. A bar graph is good here (a box plot is even better)

Hints

- Start this project early
- Keep things simple