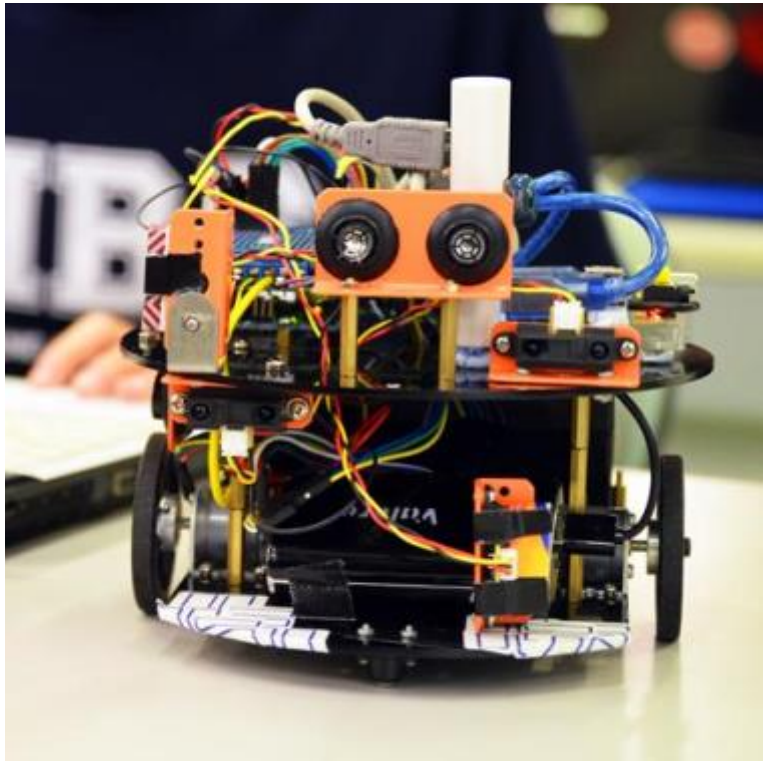
Controllers, Motors & Sensors

Ravi Suppiah



Topics

- Processing
  - Arduino
- Motion
  - DC motors + VNH-5019 board + PWM
  - Motor encoders + PID control + dead reckoning
- Sensing
  - Magnetometer
  - Sharp IR rangefinders
  - URM-37 ultrasonic rangefinders

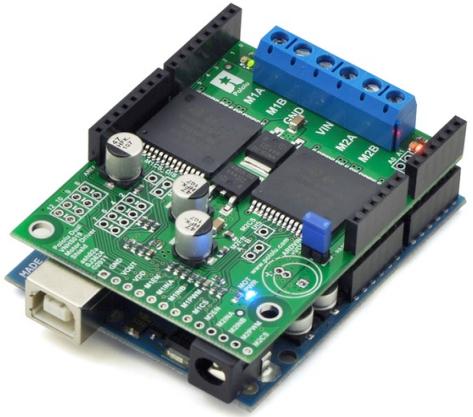
Arduino

- Open-source 8-bit micro-controller
- Easiest way to program the Arduino controllers is using the standard Arduino IDE
- Uses a subset of standard C
- Two required functions
  - setup() : This function is called once when a board starts running (after a reset or when power is applied)
  - loop() : This function is run repeatedly
- Other functions may be declared, as per C syntax
- Arduino examples: http://arduino.cc/en/Tutorial/HomePage
- Arduino C documentation: http://arduino.cc/en/Reference/HomePage

DC Motors

- DC motors are simple to use
  - Apply a voltage across both terminals and motor shaft spins
  - No polarity: reverse the voltage and motor spins in the opposite direction
  - Apply less voltage and motor spins slower
- Pololu 47:1 gearmotor (low power version)
  - 6V, 2.2A stall current
  - Quadrature encoder mounted (more on this later)

Pololu VNH-5019 motor driver



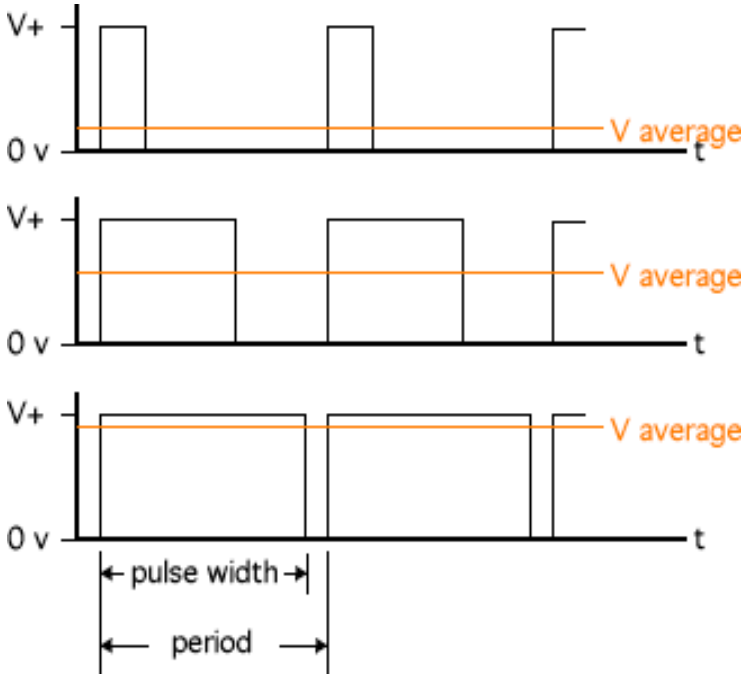
- 2.2A stall current = maximum possible drawn by motor
  - When 6V applied to terminals but shaft is stopped
- 80mA free-run current draw
  - No load applied to the motor shaft, running freely
  - Even this is too much for the Arduino to supply directly through its signal pins
  - Will cause the Arduino to reset itself
- Have to control/power motors using motor driver
  - Pololu VNH-5019 driver chip can drive 2 motors, 12A per channel
- Bonus feature: VNH-5019 shield provides regulated 5V to run Arduino off 6V battery

DC motor speed control

- Recall that increased voltage = increased speed
- Problem #1: Motors have a minimum voltage threshold
  - Friction in the bearings, etc
  - Motors will not start spinning below this minimum voltage (~1V on the 6V Pololu motors)
- Problem #2: Motor torque is dependent on voltage
- Increasing/decreasing voltage directly is a poor method for controlling DC motors

Pulse Width Modulation (PWM)

- Big idea: Instead of varying voltage, vary duty cycle
- Essentially, switch the motor on and off very fast
  - 20% duty cycle = motor on 20% of the time
  - 80% duty cycle = motor on 80% of the time
  - If the period is short enough, the effect is like increasing/decreasing the voltage applied
- But with PWM, the motor develops full torque during the ON portion
  - With PWM and 50% duty cycle, full torque developed 50% of the time
  - When applying 50% voltage, only half torque developed



VNH-5019 + PWM

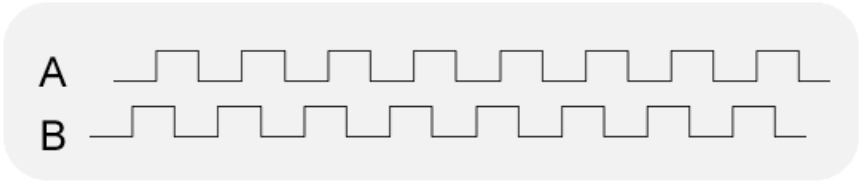
- VNH-5019 watches PWM inputs and applies full power to the motor when the PWM signal is high, cuts power when PWM signal is low
- Pinouts for the VNH-5019 found in datasheet provided
  - Direction control pins
  - Motor PWM pins
- analogWrite(motorPin, value)
  - 20% speed: analogWrite(motorPin, 0.2*255);
  - 100% speed: analogWrite(motorPin, 255);
- More details in example Arduino sketches

Odometry with motor encoders

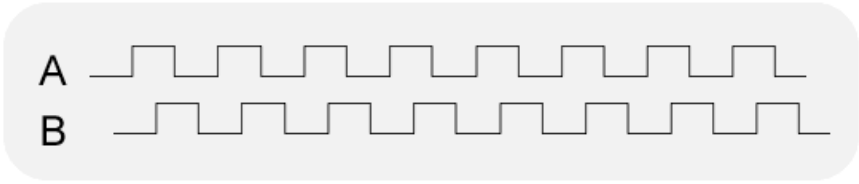
- PWM allows us to control the speeds at which the motors rotate
- How do we know how far they've actually turned?
  - Need some form of feedback
  - From number of rotations completed + size of wheel, can calculate distance travelled
- Encoders provide rotational feedback about the position of the motor shaft
- Odometry reading
  - Calculate how many revolutions the motor has completed
    = how many rotations the wheel has completed
    = what distance that wheel has travelled
- Not perfect - depends on wheels not slipping too much, but good enough

Motor encoders

- Encoders on Pololu motors use Hall effect (magnetic) sensors (details are not important)
- Two channels of output: A and B
  - Sensors arranged so that channels are 90 deg out of phase
  - Quadrature encoders
- Using two channels allows us to determine speed and direction
  - Suppose B leads A = clockwise rotation
  - Then if A leads B = counterclockwise rotation
- Pololu motor encoders generate 48 “ticks” or “counts” per rotation of the motor shaft
  - Each full rotation of the ~47:1 gearbox output shaft
    = 46.851 rotations of the motor shaft
    = 2249 ticks per wheel rotation



B leads A



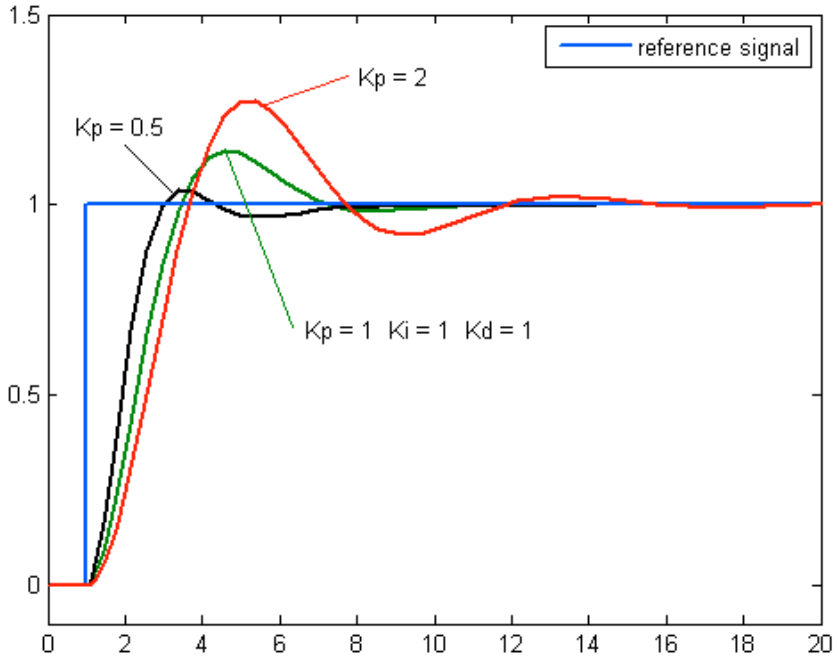
A leads B

Closed-loop speed control with encoders

- Encoder feedback doesn't just let us calculate distance travelled
- Also useful for closed-loop speed control
- Can also be used to monitor/control motor speed
  - Correct for slight differences in left/right motor output
  - Ensures robot can move in a straight line
- Suppose you have a target speed to reach
  - Monitor how many ticks have been seen in the last X milliseconds - calculate instantaneous motor speed
  - Adjust PWM duty cycle as necessary
- Question: How should the PWM adjustment be done?

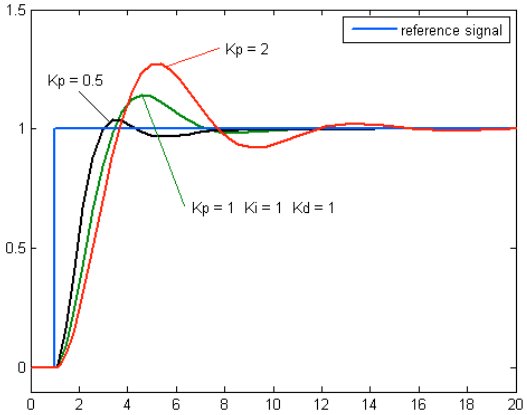
Proportional, Integral, Derivative (PID)

- A control loop feedback mechanism
  - Given a target value and a current value
  - Calculate the error and adjust control inputs to minimize the error



Proportional, Integral, Derivative (PID)

- Attempts at quick, intuitive explanations
  - Proportional term
    - The further you are from the target value, the bigger the change you should make in the control input
    - Controls speed of the system response
  - Integral term
    - The P term alone is unable to reduce errors to zero
      - Consider what happens as you get closer to the target value, and your P term correspondingly shrinks
      - Steady state is reached at an offset from target value
    - The I term sums up errors over time and eventually drives the error to zero
  - Derivative term (not always used)
    - If the control input is rapidly increasing, derivative term acts to slow it down

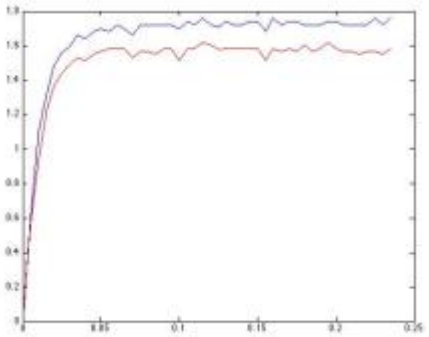


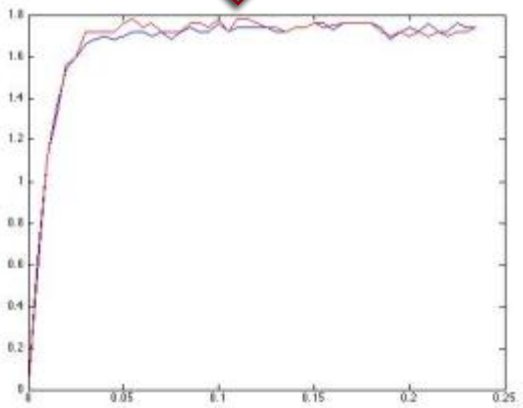
Arduino PID Library

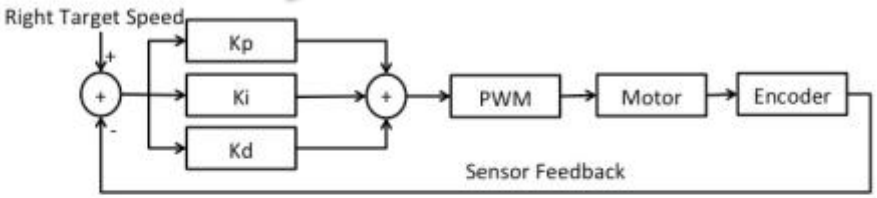
- PID library for Arduino
  - http://playground.arduino.cc/Code/PIDLibrary
- Theory and tutorials for Arduino PID Library
  - http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/
- Instructions for installing/using Arduino libraries
  - http://arduino.cc/en/Guide/Libraries

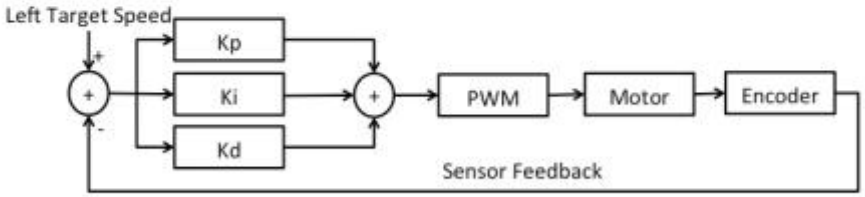
Analysis & Implementation

- Performance before PID
- Implementing PID
- Performance after PID



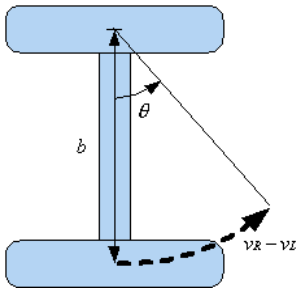


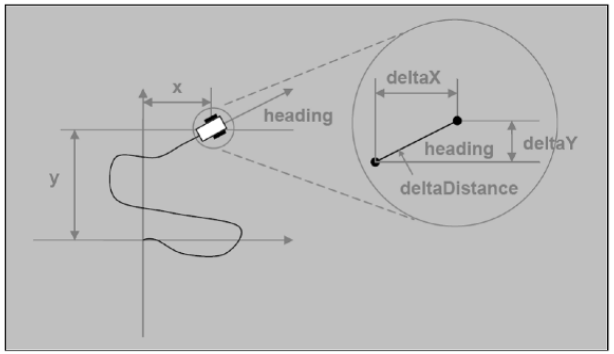




Dead Reckoning

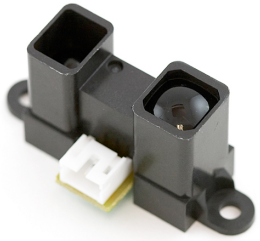
- Dead reckoning
  - Estimating position based on previous position + movement
  - Without using external reference
- Possible if you know how far you moved in the last X seconds
  - Encoder feedback
- Theory and derivation of equations too long to be covered here. See:
  - http://rossum.sourceforge.net/papers/DiffSteer/DiffSteer.html
  - http://courses.essex.ac.uk/ce/ce215/CE215%20slides/Differential%20Driving%20and%20Dead%20Reckoning.pdf

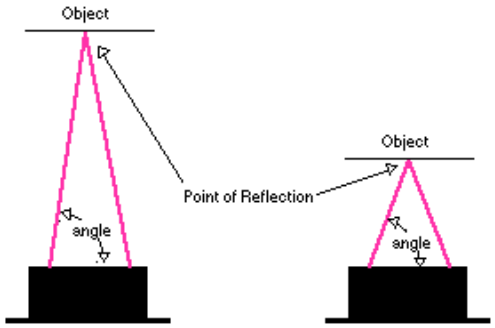




Sharp IR rangefinders

- IR emitter + linear detector
- Single IR spot emitted
  - At 40cm, approx 5mm diameter
  - At 1m, approx 10mm diameter



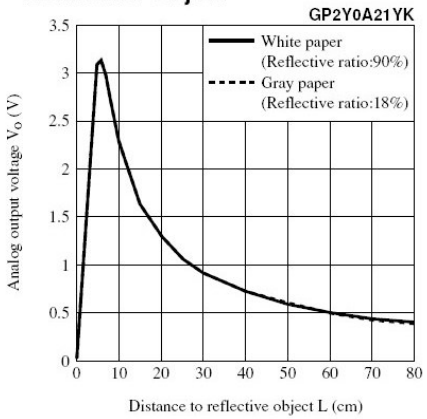


- Lens in housing projects image onto linear detector
  - As object distance increases, projection of IR spot shifts toward the emitter location
- Very important:
  - All Sharp IR sensors based on this design have a minimum distance
  - 10-80cm, 20-150cm, etc
  - Do not work well on reflective and transparent surfaces

Calibrating the Sharp IR sensor

- Sensor returns an analog reading (a voltage) that needs to be converted into a distance reading
- Sensor response is non-linear and discontinuous
- Guide to linearizing the sensor output
- http://www.acroname.com/robotics/info/articles/irlinear/irlinear.html

Fig.5 Analog Output Voltage vs. Distance to Reflective Object

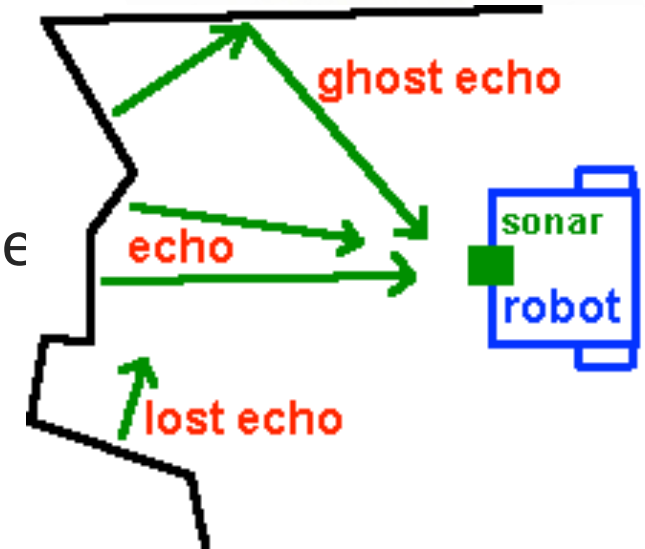


Scanning Rangefinders

- Modify setup to return multiple readings for X angle of view
  - Mount IR rangefinder on a servo motor
  - Rotate servo motor from 0 through 180 deg in 10 deg steps
    - At each position, get a rangefinder reading
  - Result is an array of polar readings for entire 180 deg ahead
- http://www.societyofrobots.com/sensors_sharpirrange.shtml
  - See section: Range Finder Basic 2D Mapping
- Servo motor:
  - http://www.engineersgarage.com/articles/servo-motor
  - http://playground.arduino.cc/ComponentLib/servo

URM-37 Ultrasonic Sensor

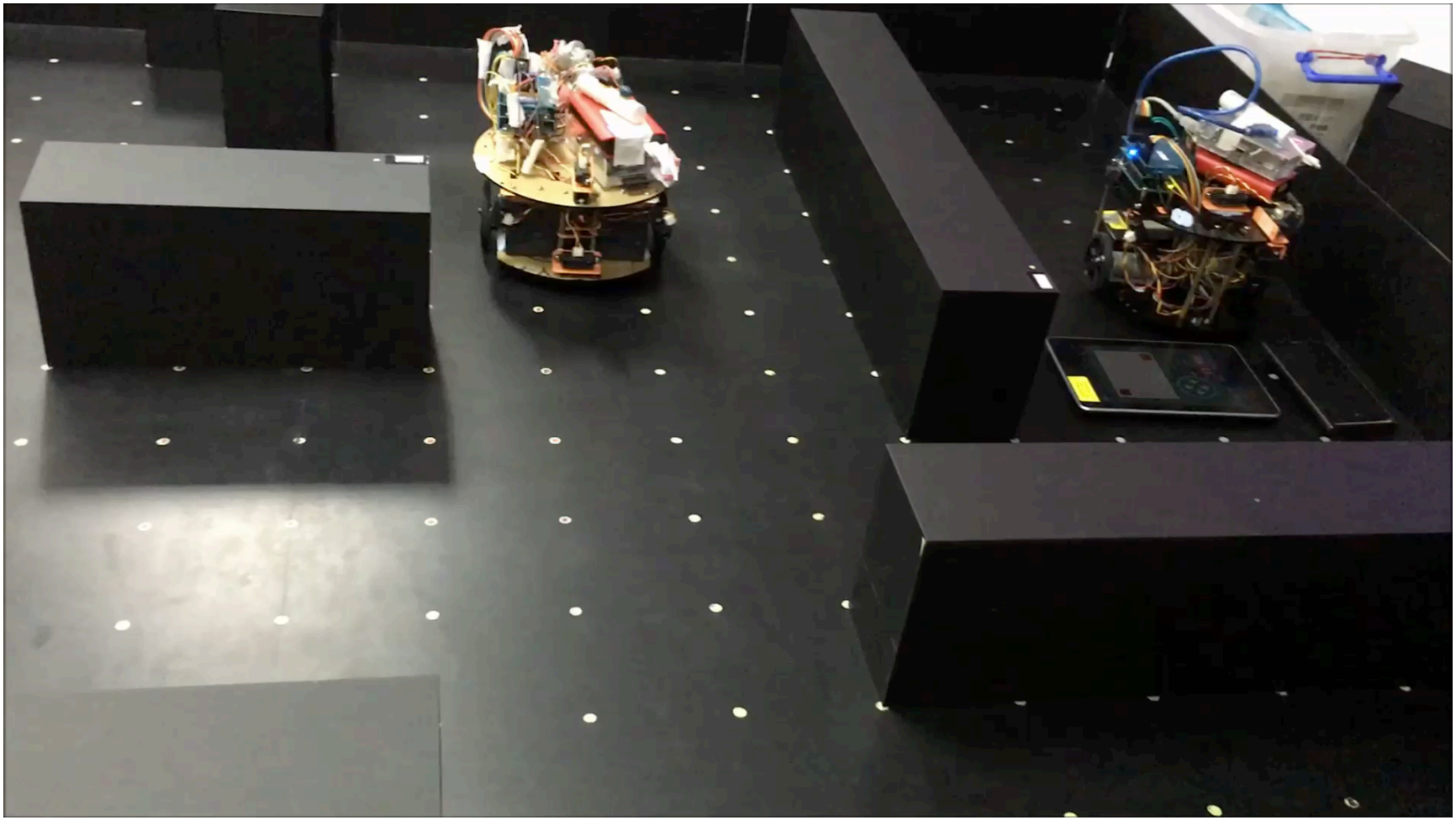
- Emit an ultrasonic pulse, wait for return
  - From time taken to receive return pulse, can calculate distance to target
  - Onboard processor does the processing and returns result over Serial, TTL, PWM
- Range of this model: 4-300cm
- Wide beam (compared to Sharp IT)
  - Need to be careful of ghost echoe



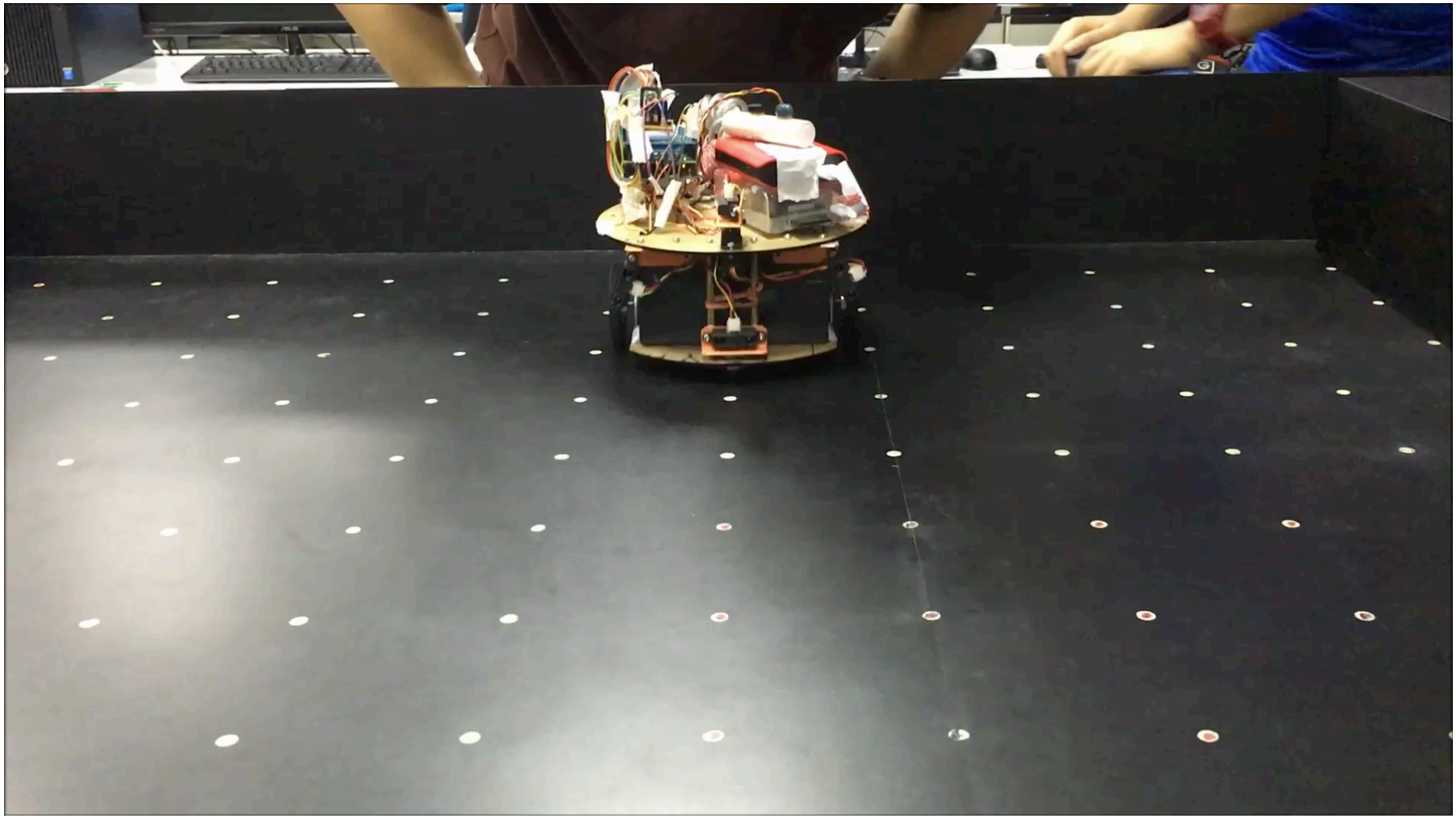
Batteries

- You are free to design and use your own Power Supply / Battery.
- More Power = Faster
- Faster != Better ??
- You must perform your calibration of sensors and motors ONLY when batteries are (FULLY) CHARGED!

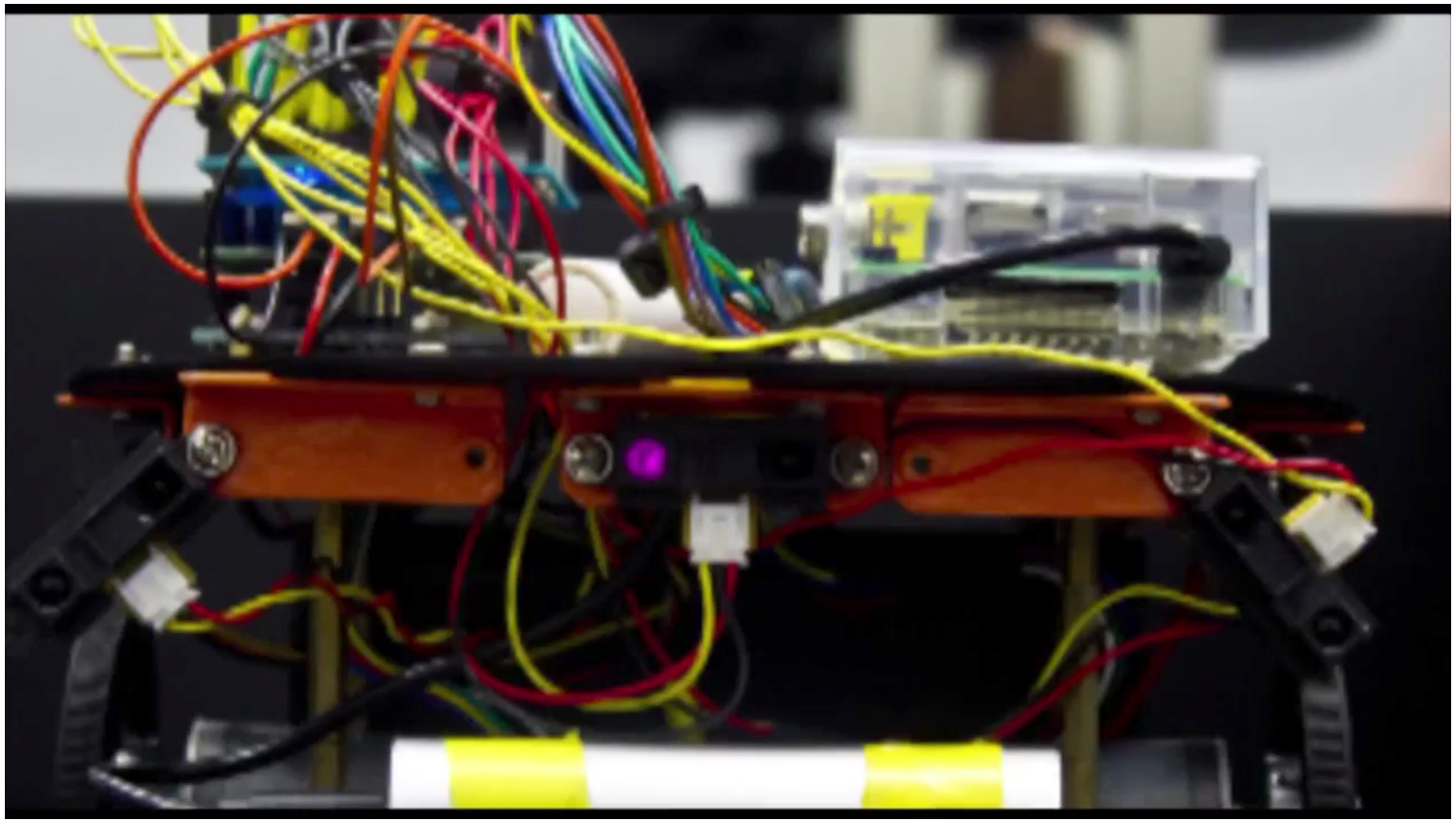
Going Straight?



Going Straight!



Motor Control & Calibration



Do your own work!

- As tough as it gets, do remember to do your own work.
- It’s ok to ask around for advise and guidance but Outsourcing your work to someone else is strictly NOT allowed.

Actual Challenge

- Now let's see what the actual challenge is all about…