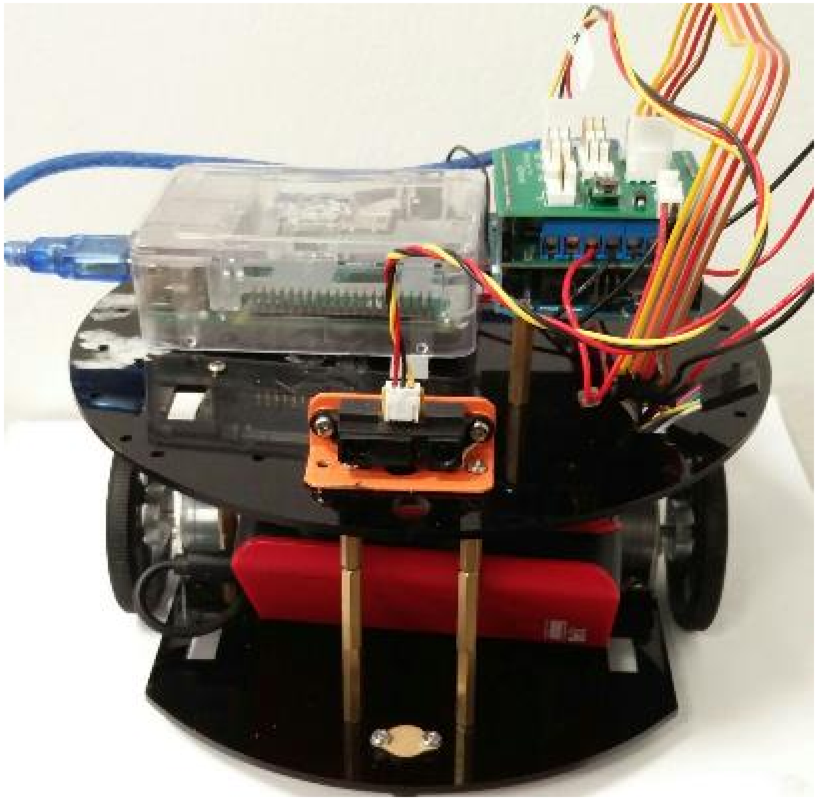
Controllers, Motors & Sensors

Smitha K G



Topics

- Hardware layer
  - Arduino uno
  - DC motor (Pololu low power 6V)
  - Motor driver shield (VNH5019)
  - Battery (6V Unicell lead acid battery)
  - Proximity sensor (Long (GP2Y0A02YK)and short range (GP2Y0A21YK))
  - URM-37 ultrasonic rangefinders (SONAR)
  - MDP power regulator/interface board

Arduino

- Open-source 8-bit micro-controller based on Atmega328
- Easiest way to program the Arduino controllers is using the standard Arduino IDE
- 14 digital input/output pins and 6 analog pins
- Powered using USB-B cable
- 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
- Arduino examples: http://arduino.cc/en/Tutorial/HomePage
- Arduino C documentation: http://arduino.cc/en/Reference/HomePage

DC Motors

- Pololu DC motor (https://www.pololu.com/product/2285)
- 6V (max voltage) →the motor spins at the rate of 120RPM (max allowed rpm)
- The wheel provided is the Pololu 60x8mm wheel (https://www.pololu.com/product/1420) with diameter of 6cm
- Pololu 47:1 gearmotor (low power version) 6V, 2.2A stall current
- Testing motors
  - Apply a voltage across both terminals and motor shaft spins corresponding to voltage (max 6V)
  - No polarity: reverse the voltage and motor spins in the opposite direction
  - Apply less voltage and motor spins slower

6 V	low-power **(LP)**	2.4 A	6200 RPM	2 oz-in	1:1 LP 6V w/encoder	
			1300 RPM	8 oz-in	4.4:1 LP 6V w/encoder	4.4:1 LP 6V
			590 RPM	17 oz-in	9.7:1 LP 6V w/encoder	9.7:1 LP 6V
			290 RPM	33 oz-in	20.4:1 LP 6V w/encoder	20.4:1 LP 6V
			170 RPM	50 oz-in	34:1 LP 6V w/encoder	34:1 LP 6V
			120 RPM	65 oz-in	47:1 LP 6V w/encoder	47:1 LP 6V
			78 RPM	95 oz-in	75:1 LP 6V w/encoder	75:1 LP 6V
			58 RPM	130 oz-in	99:1 LP 6V w/encoder	99:1 LP 6V
			34 RPM	200 oz-in	172:1 LP 6V w/encoder	172:1 LP 6V
			25 RPM	220 oz-in	227:1 LP 6V w/encoder	227:1 LP 6V
			15 RPM	300 oz-in	378:1 LP 6V w/encoder	378:1 LP 6V
			11 RPM	400 oz-in	499:1 LP 6V w/encoder	499:1 LP 6V

Encoder

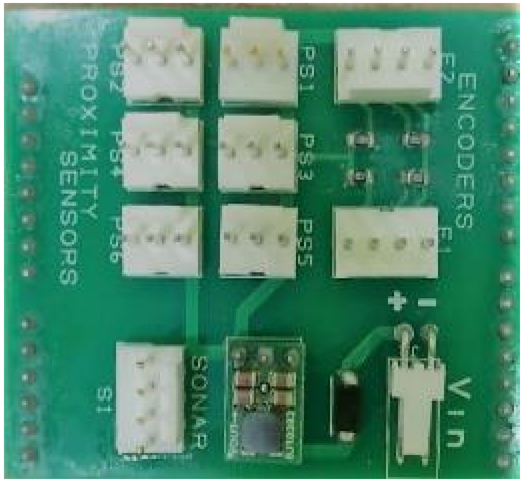
- Integrated 48 CPR quadrature encoder on the motor shaft. (https://www.pololu.com/product/2285)
- It provides 2248.86 counts per revolution of the gearbox’s output shaft (needed to calculate the set speed)
- The encoders use a two-channel Hall effect sensor to detect the rotation of a magnetic disk on a rear protrusion of the motor shaft.

Pololu VNH-5019 motor driver

- https://www.pololu.com/product/2502
- https://www.pololu.com/docs/0J49 (do read the manual to understand more about the mappings and pin details)
- VNH-5019 shield provides regulated 5V to run Arduino off 6V battery

Power regulator board

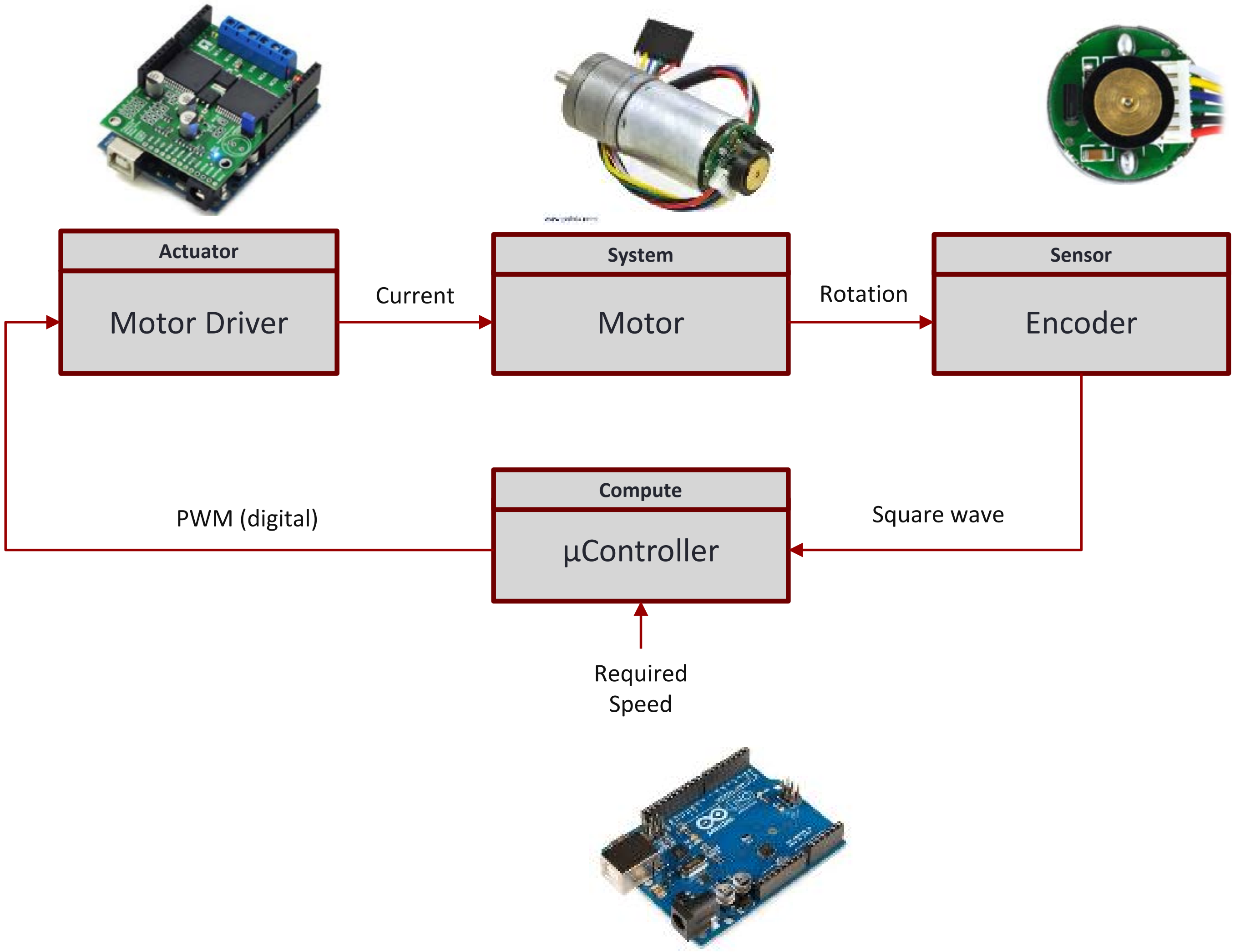
- To connect all sensors and the motor encoders
- To mount: mount the motor driver shield on the Arduino Uno and then mount the interface board on top of the motor driver shield.
- Has voltage regulated using buck/boost convertor
- To power the power regulator board: connect the ground of battery to –ve in the power pin, and positive of battery to the +ve pin



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

Speed control system

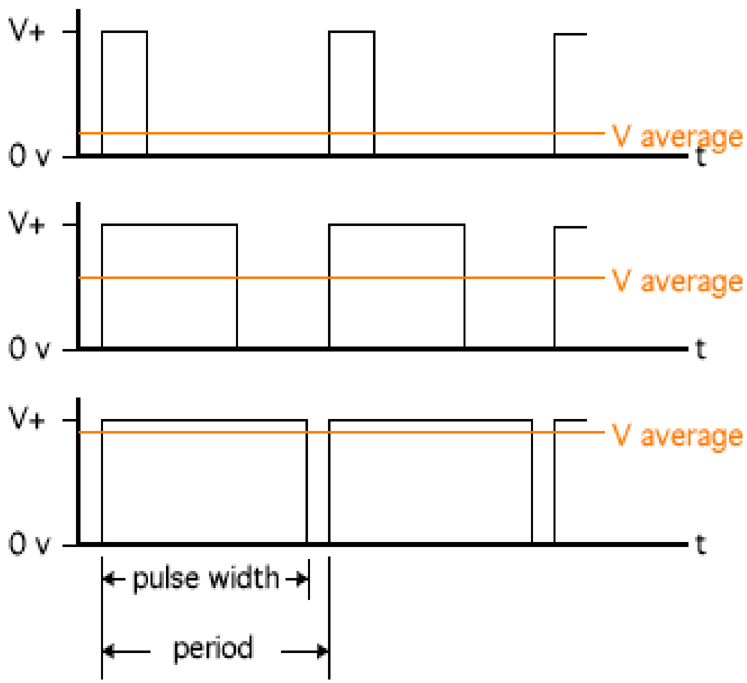


System working steps

- Microcontroller board drives the motor through PWM
- In the board library, required Motor Speed is specified by (u) and is in the range of -400 to +400.
- Driver sends corresponding current to motor to drive it
- PWM controls the current to the motor and thus the rpm of the motor.

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
  - If the period is short enough, the effect is like increasing/decreasing the voltage applied



VNH-5019 + PWM

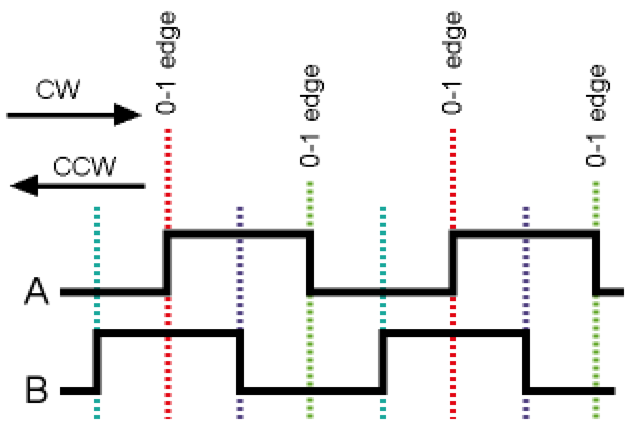
- VNH-5019 watches PWM inputs and applies full power to the motor when the PWM signal is high, cuts power when PWM signal is low
- Pinouts for the VNH-5019 found in datasheet provided
  - Direction control pins
  - Motor PWM pins
- analogWrite(motorPin, value)
  - 20% speed: analogWrite(motorPin, 0.2*255);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

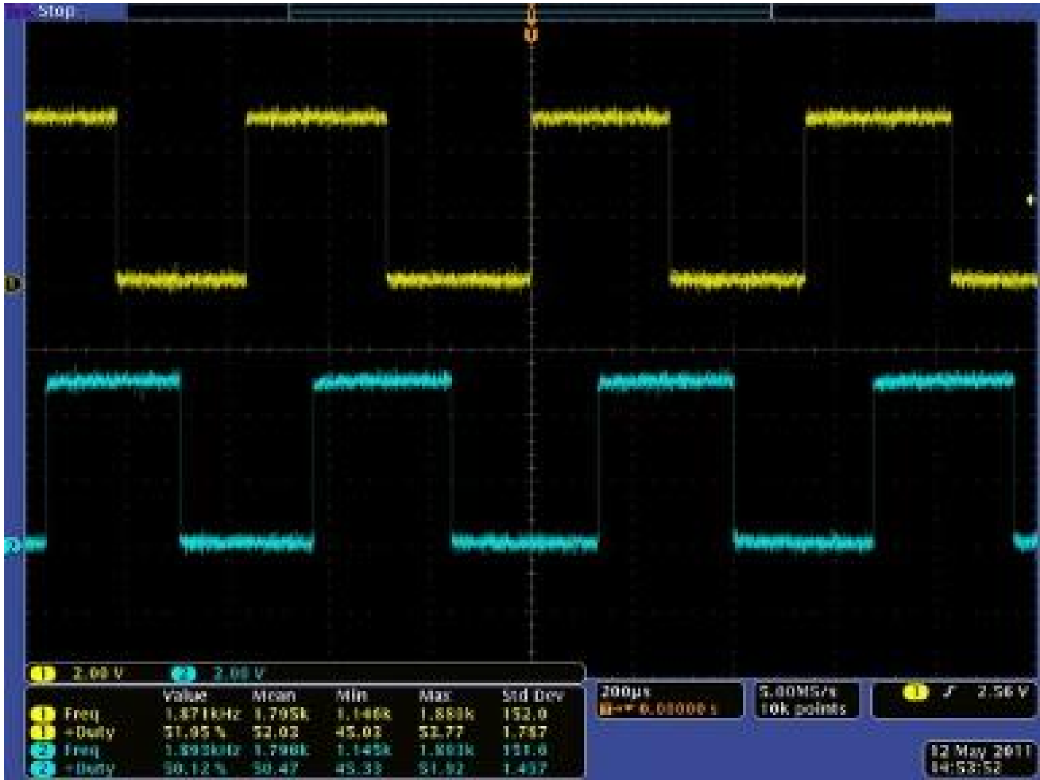
Motor encoders

- Encoders on Pololu motors use Hall effect (magnetic) sensors
- 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 = 47.851 rotations of the motor shaft = 2249 ticks per wheel rotation



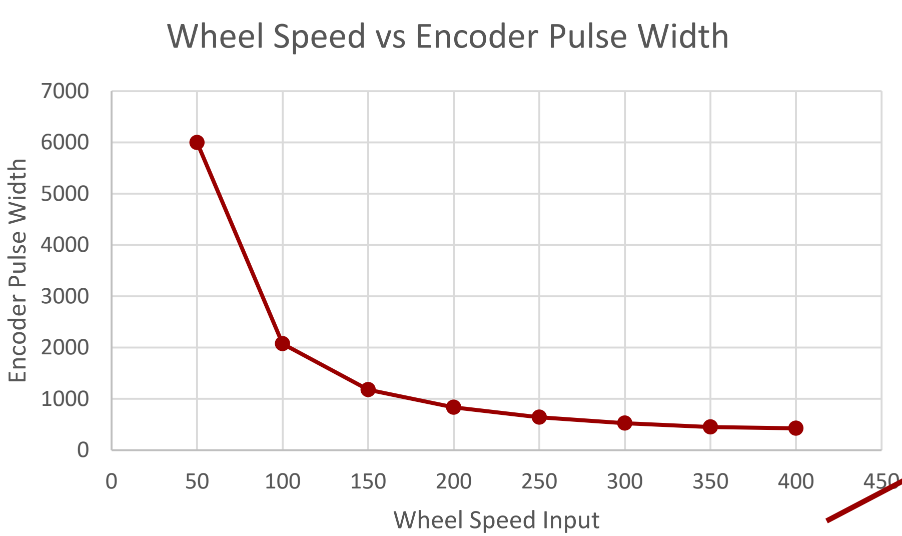
Encoder

- The A and B outputs are square waves from 0 V to Vcc approximately 90° out of phase.
- The frequency of the transitions tells you the speed of the motor, and the order of the transitions tells you the direction.
- The following oscilloscope capture shows the A and B (yellow and white) encoder outputs using a motor voltage of 6 V and a Hall sensor Vcc of 5 V:



Encoder

- Encoder is used to measure the speed of motor
- Need to convert square wave from motor encoder to a meaningful speed
- Using time-width of pulse is one way -> faster the wheel speed, shorter the time-width :



Wheel speed is set speed from Arduino

- Steps to convert square-wave to wheel rpm:
  - First, read the encoder signal of each motor into any unused digital input pin of Arduino. You may use either output A or output B of each encoder.
  - Compute the time-width of the square-wave using pulseIn(pin,HIGH) but this will limit the execution time of the loop. Alternatively, interrupt can be used to calculate the number of ticks.
  - Convert time-width to wheel rpm using the fact that there are 562.25 square waves for every revolution of wheel
  - Test the calculated rpm for different motor speeds.

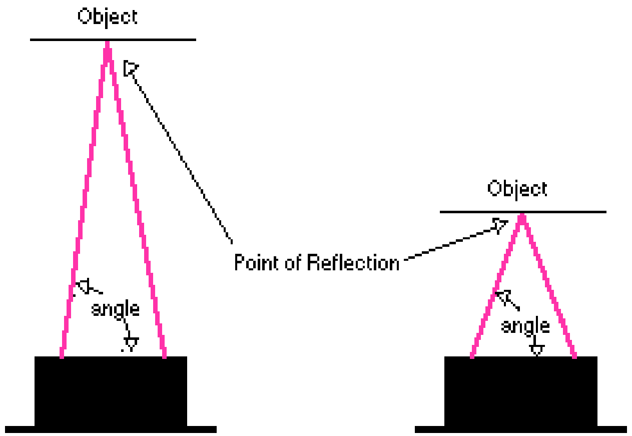
http://playground.arduino.cc/Main/RotaryEncoders#Waveform
https://github.com/PaulStoffregen/Encoder

Table – Input Speed vs Measured RPM (example)

Speed	RPM
400	131.863
350	122.2386
300	103.6323
250	84.4372
200	65.01241
150	46.05117
100	27.73403
50	9.67545
0	0

Sharp IR rangefinders

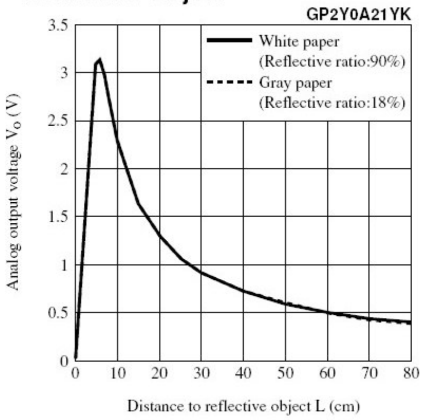
- IR emitter + linear detector
- Single IR spot emitted
- Proximity sensor: The short range proximity sensor (https://www.sparkfun.com/datasheets/Components/GP2Y0A21YK.pdf) for range 10 cm-80 cm
- long range proximity sensors (https://www.sgbotic.com/products/datasheets/sensors/GP2Y0A02YK.pdf) with range 20cm-150cm.
- Proximity sensors also have a 3pin cable with them.
- Do not work well on reflective and transparent surfaces



Calibrating the proximity 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
- https://ericjformanteaching.wordpress.com/2013/04/24/how-to-make-a-sharp-ir-sensor-linear/

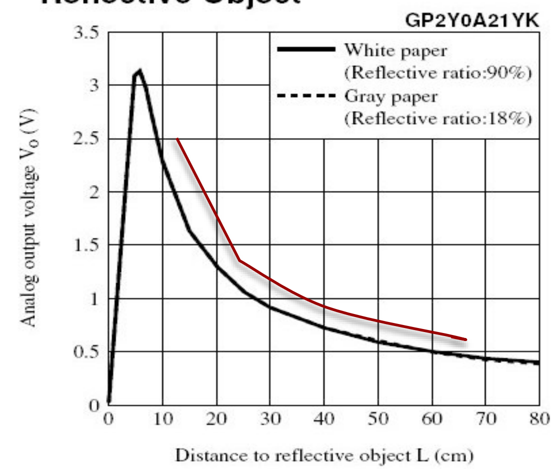
Fig.5 Analog Output Voltage vs. Distance to Reflective Object



Calibrating the proximity sensor

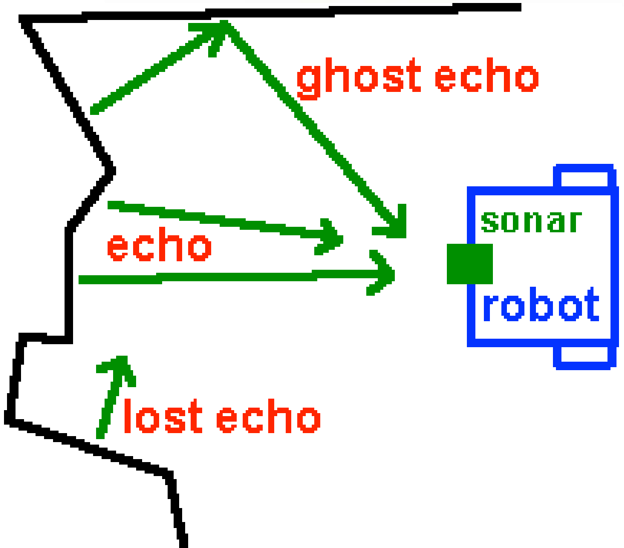
- Need to use analog filters for averaging (median of n values, mean or weighted average filters)
- Need to have a curve fitting solution
- Need to think about the placement of the sensors on the robot

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

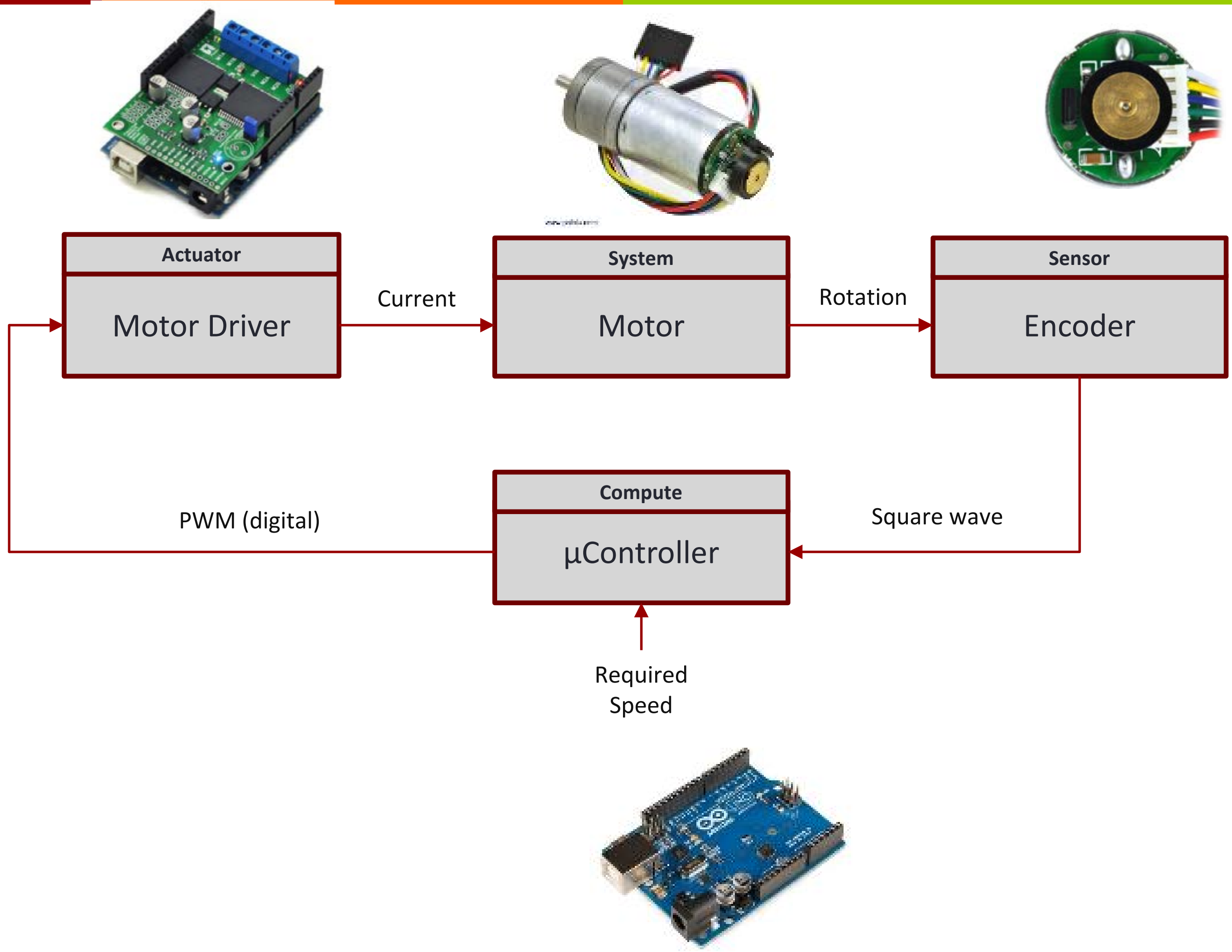


URM-37 Ultrasonic Sensor

- 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 echoes

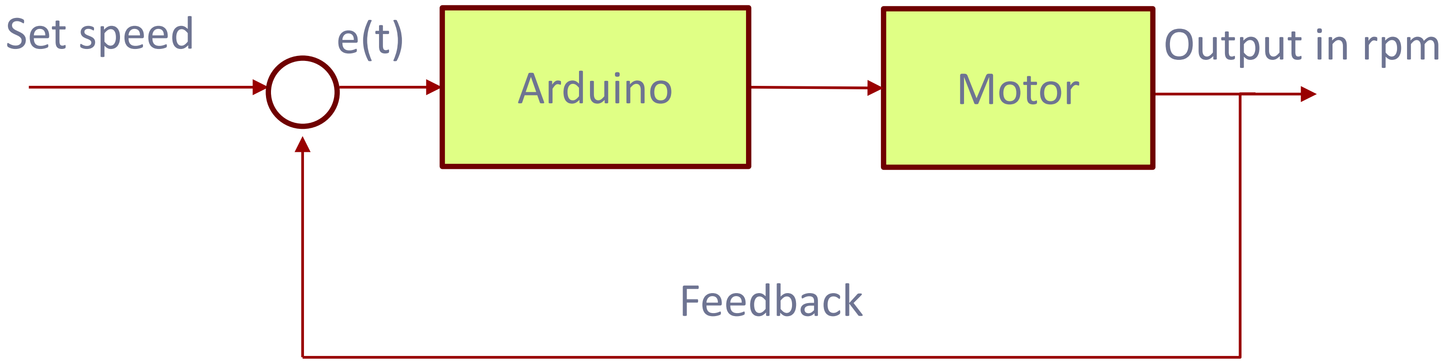


Speed control system



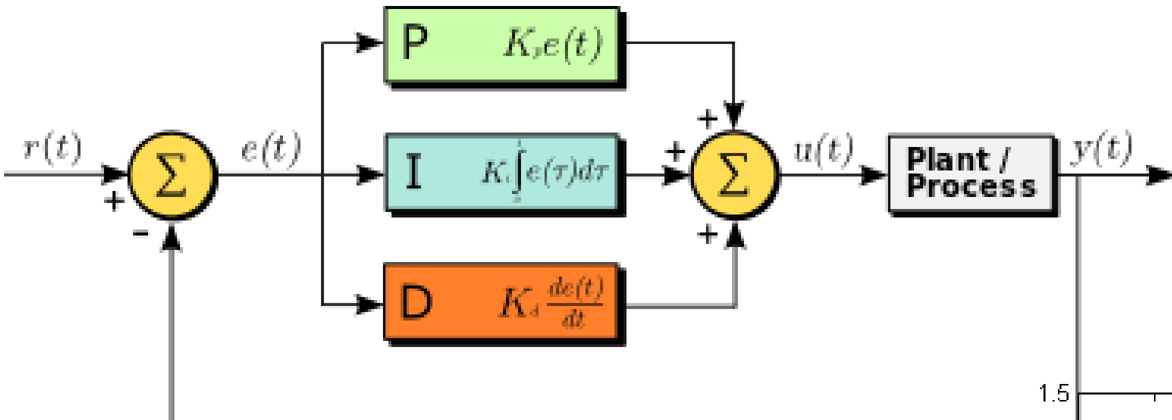
Is that exactly straight?

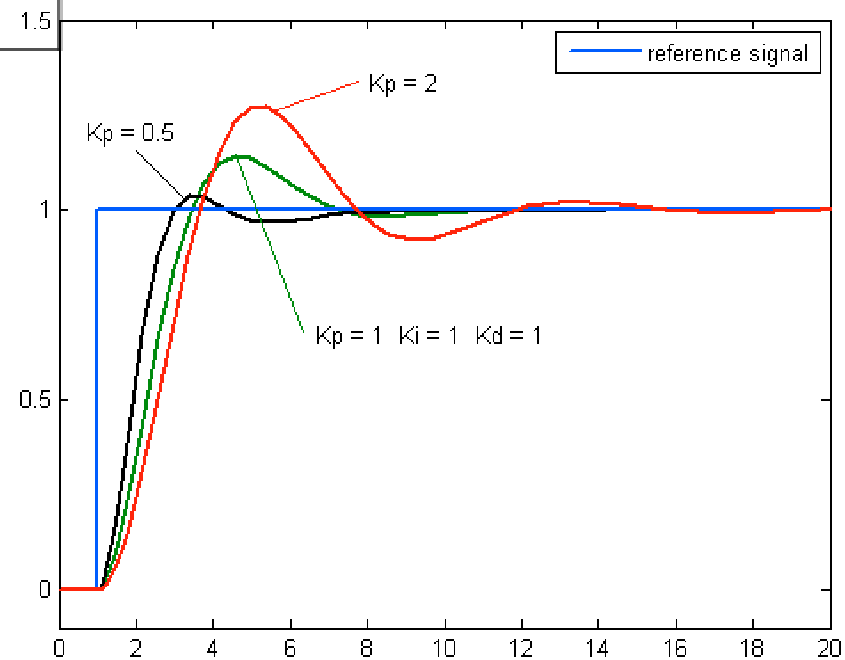
Controller



- Open loop controllers will not work due to friction, different motor characteristics etc.
- Error correction mechanism using closed loop system
- Error (e(t))= set speed-feedback
- A controller calculates the error and adjust control inputs to minimize the error

Proportional, Integral, Derivative (PID)



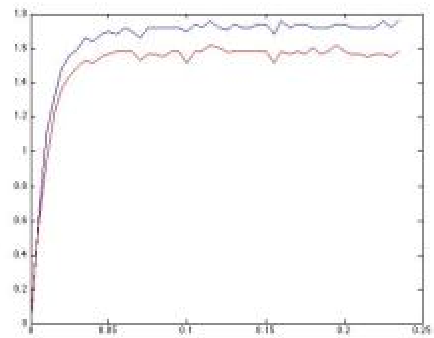


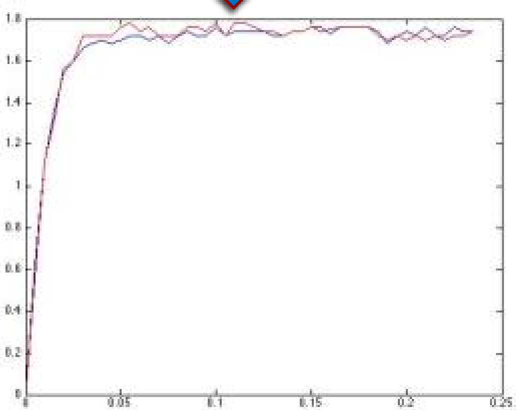
$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{de(t)}{dt},$$

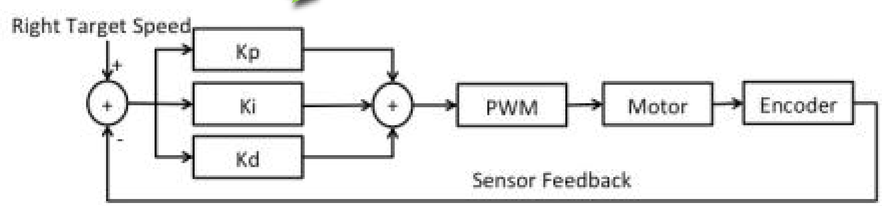
Where Kp, Ki and Kd, all non-negative, denote the coefficients for the proportional, integral, and derivative terms, respectively (sometimes denoted P, I, and D).

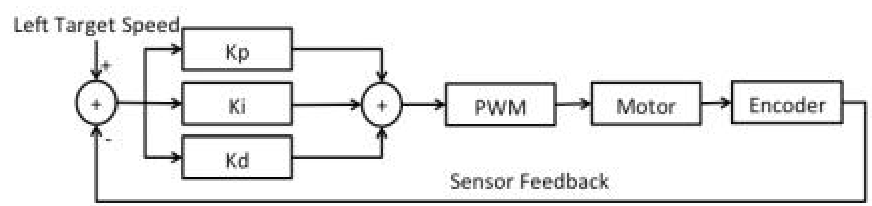
Analysis & Implementation

- Performance before PID
- Implementing PID
- Performance after PID

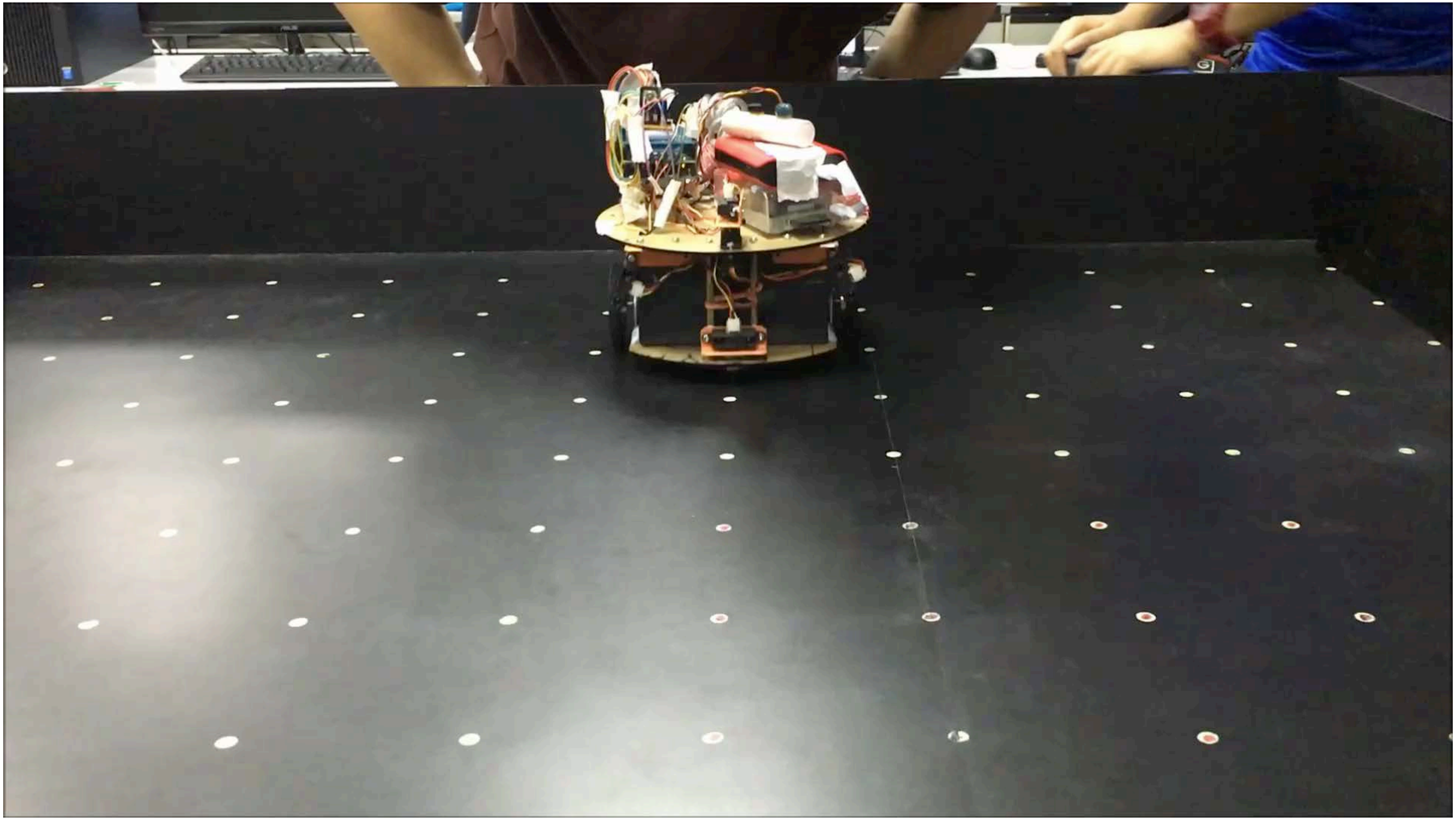




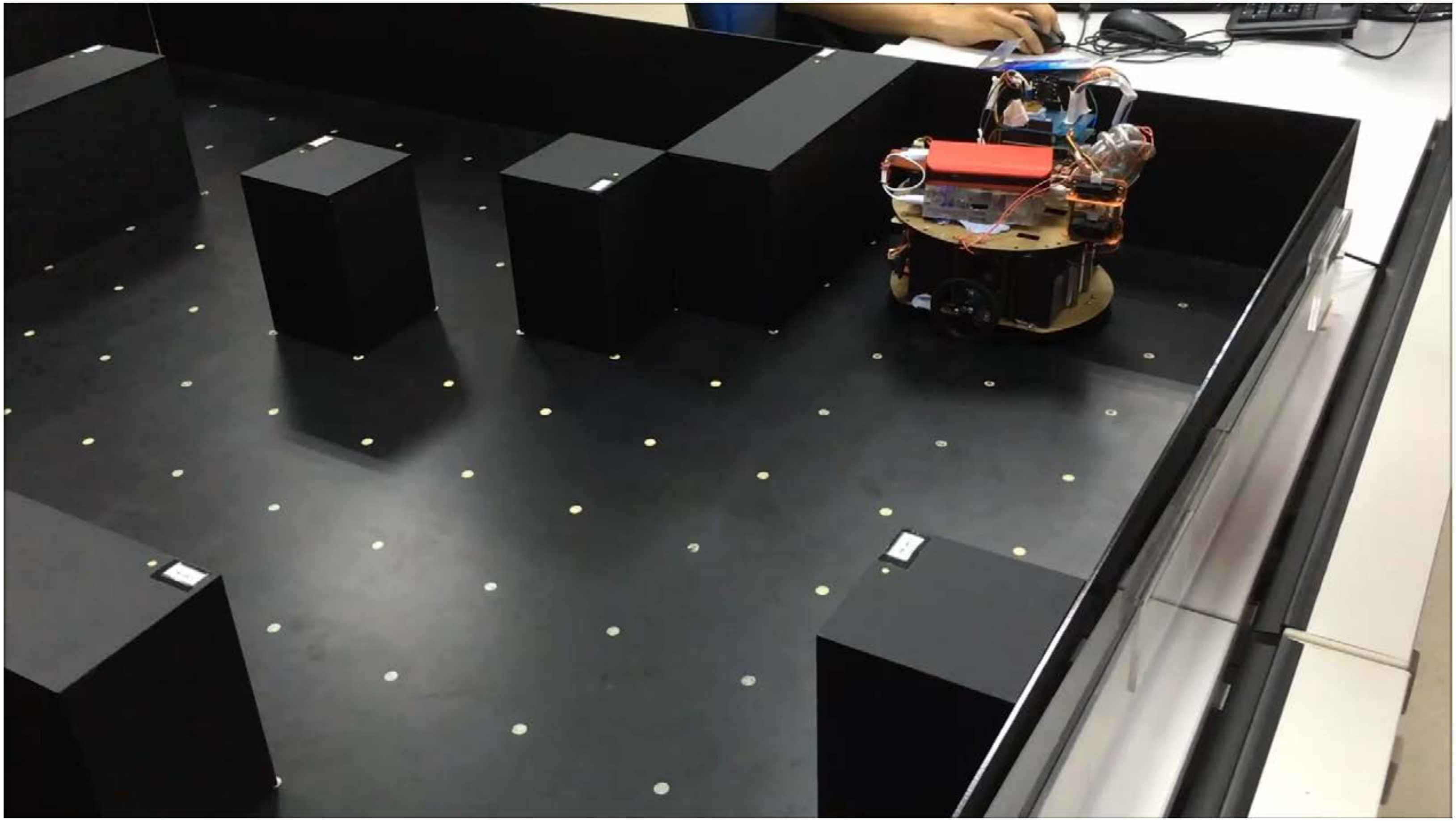




Going Straight!



Rotation



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.