EduMouse
The Trainer from Taiwan

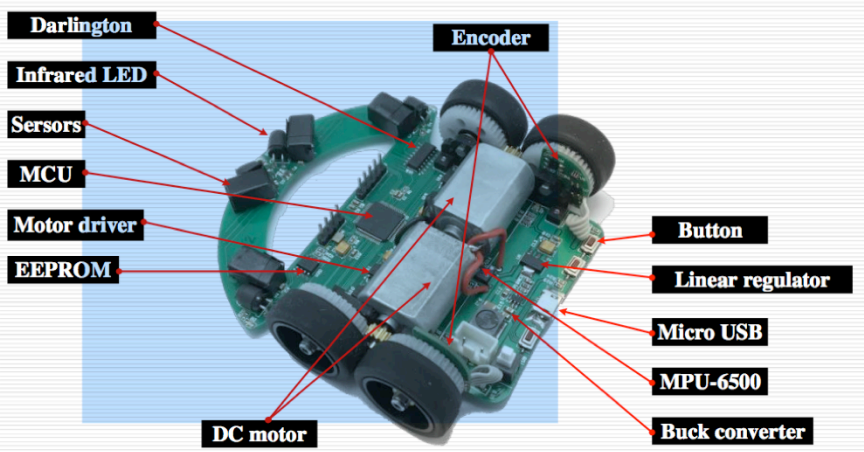
MINOS 2018

Peter Harrison

Inexpensive Training Robot

- Where can you save money making a robot like a micromouse?
  - PCB
  - Components
  - Peripherals
  - Sensors
  - Batteries
  - Wheels
  - Mechanicals
  - Motors
  - Odometry

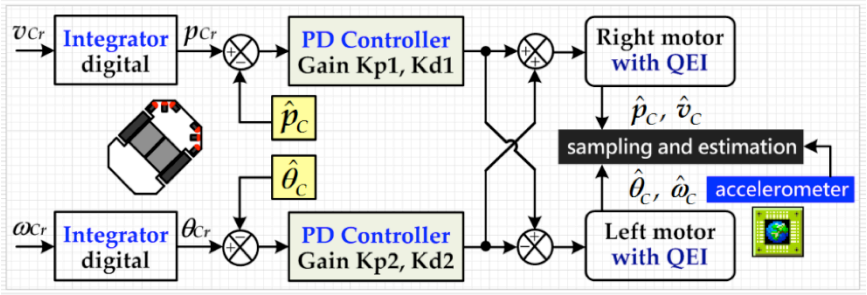
EduMouse



from EduMouse documentation – Lunghua University, Taiwan

EduMouse - Control

- ✓ **Use angular command to correct attitude errors**
- ✓ **1ms control period;**



from EduMouse documentation – Lunghua University, Taiwan

The controller system for EduMouse is common to any number of differentially-steered wheeled, mobile robots

Thus, the results are applicable to a wide variety of physical platforms

Cheap motors

- One motor for D5 cost the same as EduMouse
- But:
  - Possible variation in parameters.
  - May be inefficient.
  - Mounted examples need 0.6V to get moving.
  - Others may need 1.5V or more.
  - Cogging effects due to construction
  - Torque Ripple in operation
  - No encoder

Motors are one of the most significant ways to achieve cost savings in a micromouse.

There are some potential pitfalls to consider but these are not generally overwhelming.

Low Cost Encoders

- Magnetic encoders (AS5045 etc)
  - £12 each
  - Careful setup
  - good resolution
- Optical Reflectors
  - £0.20-£0.50 each
  - Easy setup
  - Poor resolution
- Edumouse encoders generate 24 counts per wheel rotation.
  - Wheel diameter approximately 25mm.
  - Left and right wheels averaged for position.
  - Thus, resolution is approximately 1.6mm
  - Or, about 112 steps per cell in the maze.
  - 1 step per ms implies 1600mm/s

Odometry is the key to everything.

The robot must know where it is on the ground.

A gyro can take care of all rotational movement but forward motion cannot be done solely with inertial measurement.

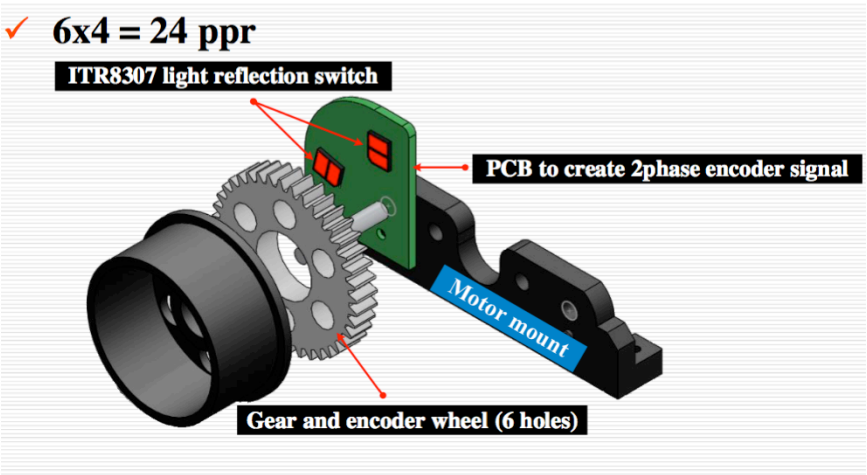
The better the encoders are, the more they will cost.

What can we achieve with the lowest possible cost?

(Note that we could use just one reflector per wheel and forego quadrature − see Kojimouse7)

EduMouse - encoders

- ✓ **6x4 = 24 ppr**



from EduMouse documentation – Lunghua University, Taiwan

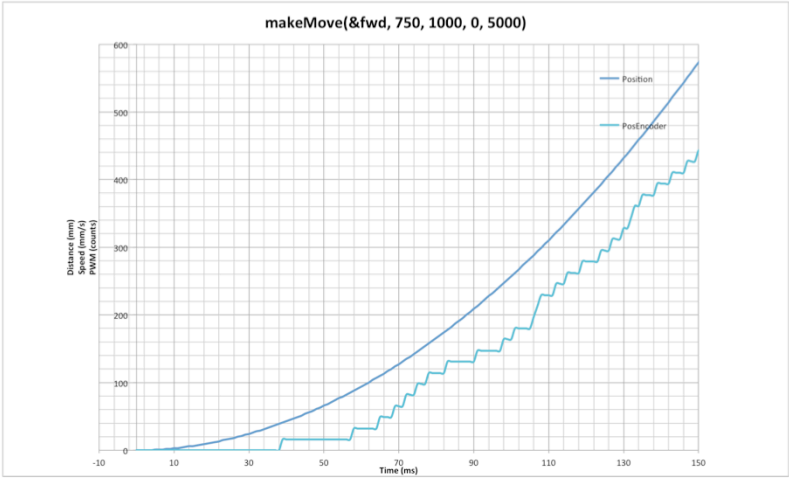
Effect of low resolution odometry on controller

- As commanded position increases, mouse appears not to move
- Controller error grows
- Motor drive increases
- Followed by a jump in the apparent position
- Controller error now negative
- Motor drive reverses polarity.

There are some significant issues relating to the use of very low resolution encoders.

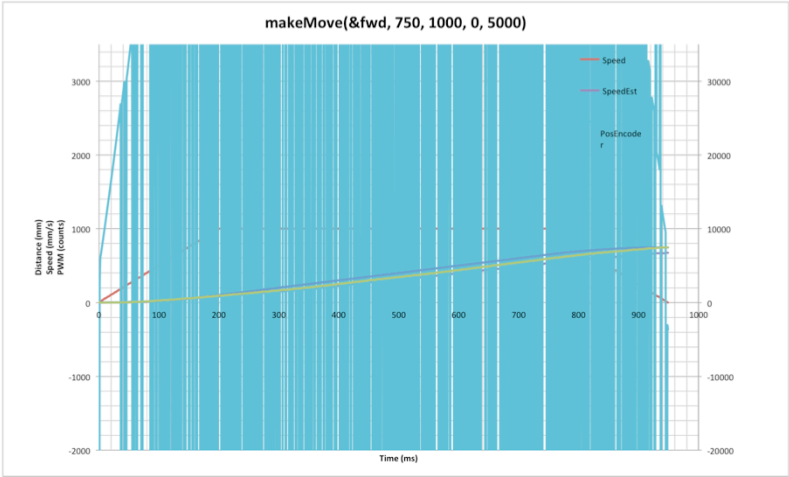
Even if overall positional accuracy is adequate, building a controller to make use of these encoders can be a challenge.

Position measurement from encoders



Using encoders only, the recorded position jumps in large steps that are not synchronous with the controller time interval

Encoder feedback only



The most simple, naïve approach is a horrible mess.

The robot will make the move but its accuracy will be poor and the wild reversals in the PWM drive put a lot of strain on the drivetrain.

And it sounds like a bucket of nails while it moves.

Estimating Position

- Overall positional accuracy probably acceptable at +/- 2mm or so.
- System needs a higher resolution estimate of position for smoother control.
- Kojima (Japan, Kojimouse) uses averaging of encoder and accelerometer readings:

  v = [encoders speed average in the last 100ms]
  + [accelerometer integral in the last 50ms]

Kojimouse7 was one of the first to use very low resolution encoders.

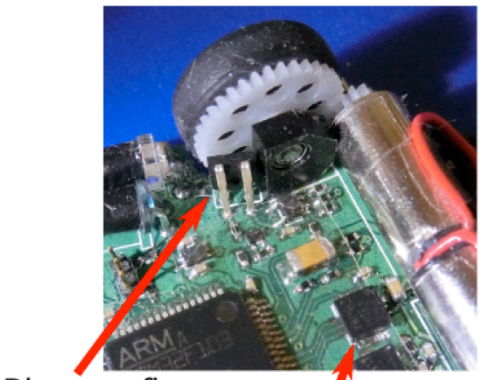
A centrally- mounted accelerometer provided information to augment the encoder data.

No quadrature here. Direction is implied by the profiler and the output from the accelerometer.

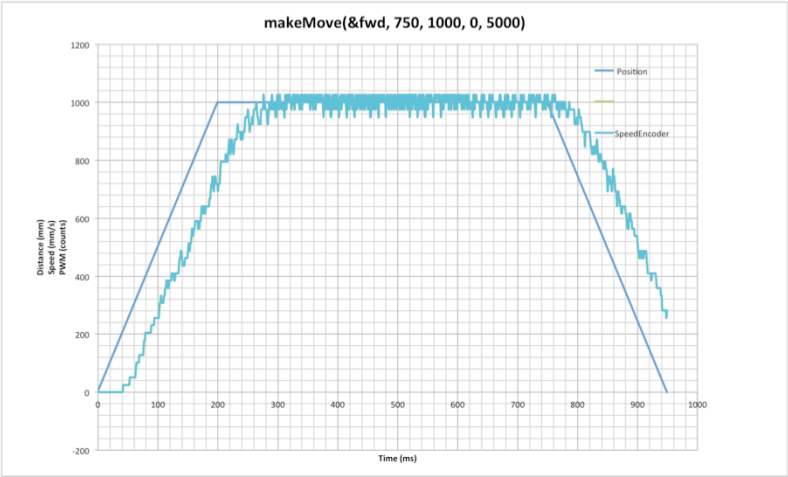
Averaging is not as expensive as you might think since the data is held in circular buffers.

The implementation is simple but a little subtle.

http://kojimousenote.blogspot.co.uk/2012/05/velocity-measurement-with-kojimouse7.html

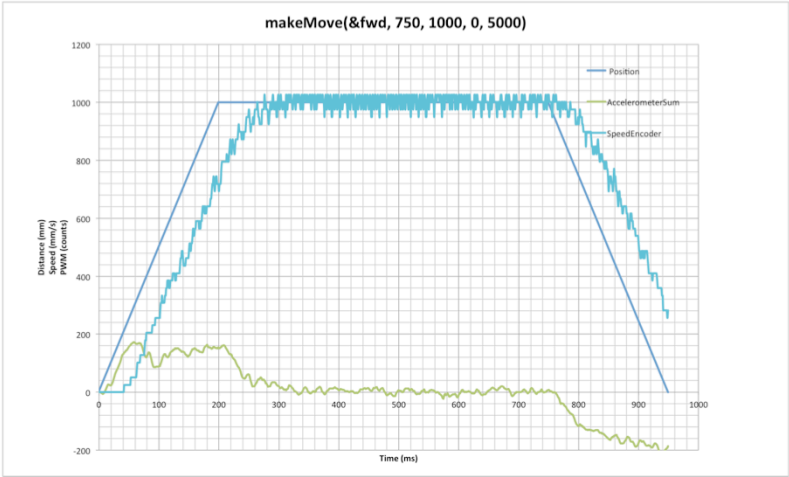


Estimate speed as average of encoder speeds over 64ms



Averaging the encoders alone effectively low-pass filters the data giving significant lag and rounding of sharp corners.

Also integrate the accelerometer over 32 ms



Minos 2018 Peter Harrison 14

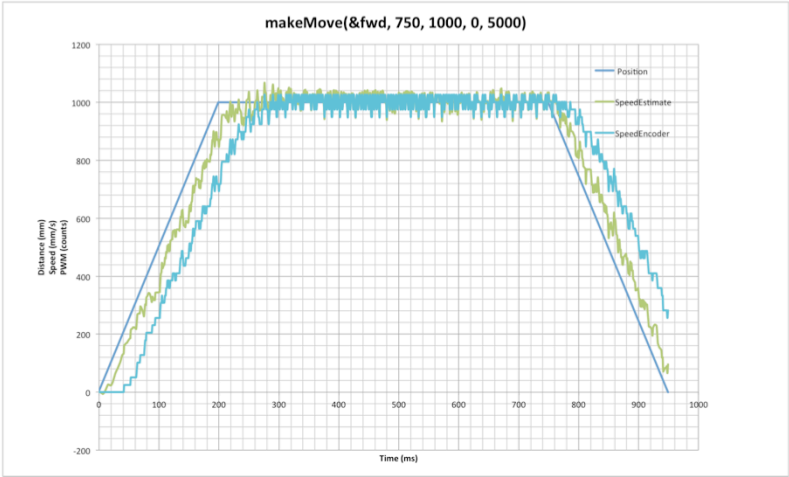
The accelerometer is very noisy but can be integrated over a short period to get an estimate of velocity.

The acceleration phase is clearly visible.

Constant velocity produces no accelerometer output.

Since the accelerometer is integrated, it is also low-pass filtered so there is no sharp transition from accelerating to constant speed.

Combine both for a better speed estimate



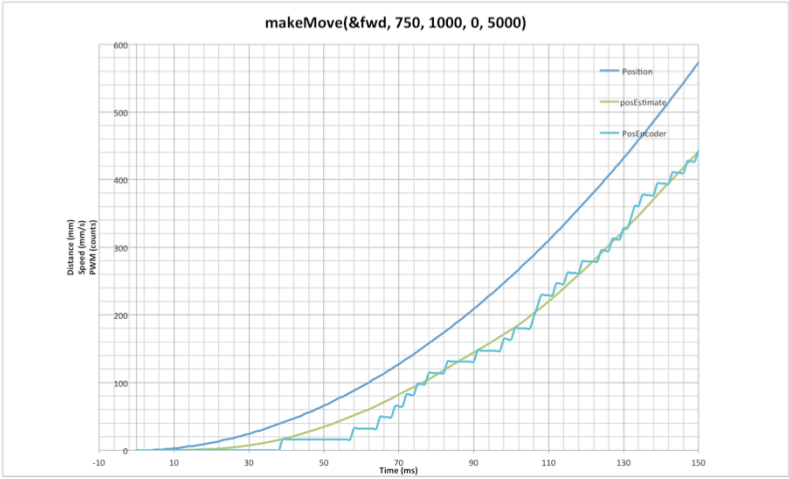
Minos 2018 Peter Harrison 15

If the accelerometer estimate of velocity is simply added to the encoder average, the overall estimate is greatly improved.

There is a tradeoff between longer averaging periods giving better estimates and shorter averages producing more timely results.

I do not know where the optimum level might lie.

Position Estimate by Kojima's method



Viwed in greater detail, it is clear that the averaging approach provides a continuous position estimate that is a good fit for the encoder data.

EduMouse State Observer

- EduMouse has a position estimation method based on a state space model of the mouse.

- Ask me after class

State space analysis is another way of looking at systems and it has numerous advantages of more classical methods.

State space observers can be used to estimate system properties where there is imperfect information available.

I do not know how to do this from first principles but I can recognise and implement a result when I see one.

A more complete description can be found in:
J. H. Su, C. S. Lee and C. W. Chen, "Sensor fusion algorithms for encoder resolution enhancement in educational mobile robots," 2016 International Conference on Advanced Robotics and Intelligent Systems (ARIS), Taipei, 2016, pp. 1-5.

State Observer Implementation

- Thankfully, it is easy to implement:
- Only two constants define the behaviour
- These are easy to determine
- Just three lines of (floating point) code

```
// position and velocity estimates from error and imu
float positionError = encoderPosition - positionEstimate;
positionEstimate += (Kv * positionError) + velocityEstimate;
velocityEstimate += (Kp * positionError) + imu.acceleration;
```


And this is the resulting implementation.

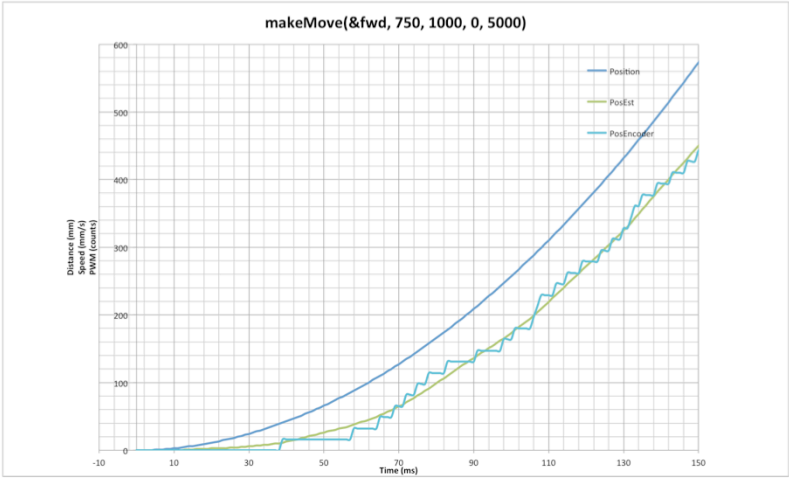
Only two numbers need to be selected and they can be determined by observation.

The natural frequency sets the response bandwidth.

The damping ratio sets the response time and overshoot as it does for any second order system.

The units here are system dependent and I need to work through the maths a little to find appropriate values in real units

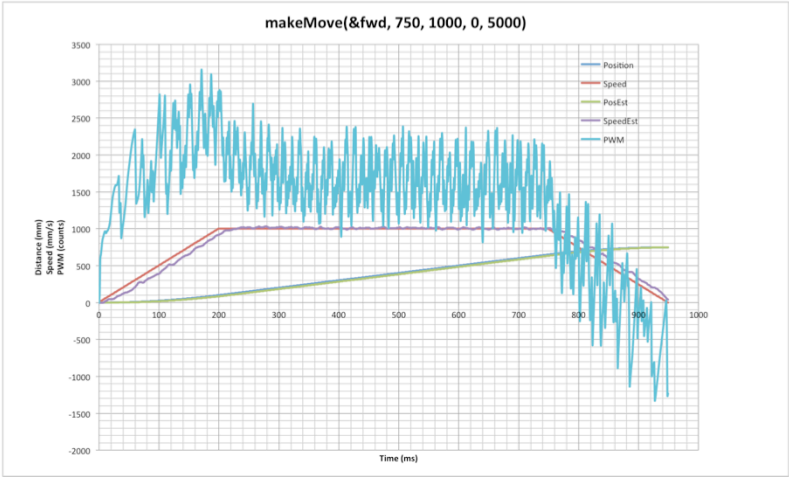
State Observer position estimate



The observer position estimate also provides a nice, continuous fit to the encoder data.

Purely by visual inspection, it looks to be a better fit than the averager used by Kojima.

Simple Move Forward



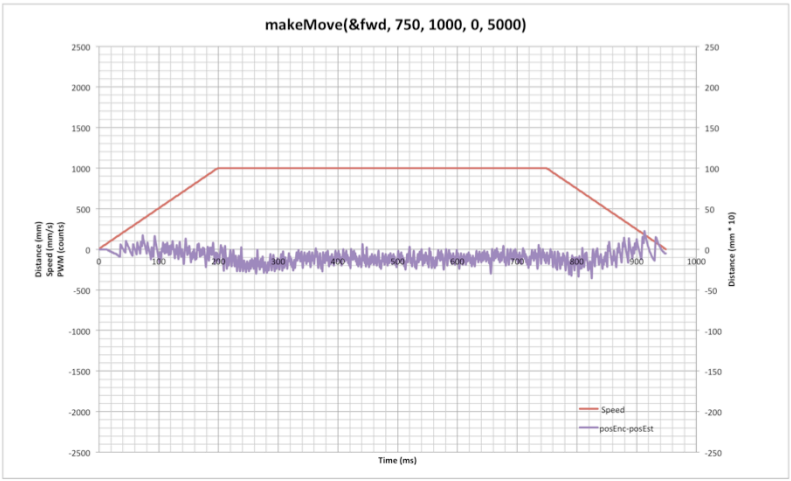
A simple move shows how the observer position estimate lets us make a pretty good control system.

The PWM drive is noisy as might be expected by the effect of a noisy accelerometer and only having an estimate of position

Also, I think there is a significant mechanical contribution from the motor and drive train.

Some tests indicate the PWM signal has strong frequency components corresponding to mechanical features in the drivetrain.

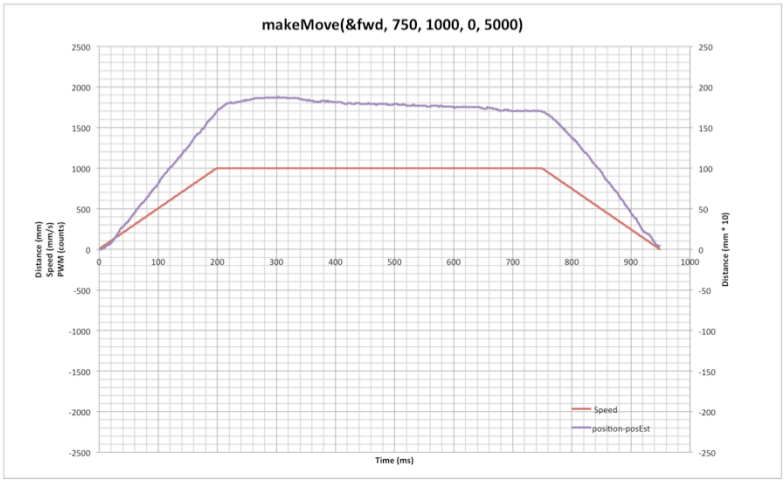
Position error (encoder-observer)



The error between the observer position estimate and the position reported by the encoders is small throughout the move but subject to quite a lot of fluctuation.

There is always an error though. Without it, there would be no drive signal to the motors.

Position error (setpoint-observer)



Most of the motor drive comes from the controller trying to reduce the error between the commanded position and the current estimate.

Unless there is an unfeasibly high controller gain, large errors are needed to generate large PWM outputs.

Remember that the motors do not even move until the PWM is up to about 10%

Effect of Accelerometer

- Accelerometer output very noisy.
- Integrating accelerometer gives velocity.
- At constant velocity, accelerometer reads zero.
- Thus, no contribution to observer estimate
- Mouse and surface tilt may cause inaccuracies
- Accelerometer should be centrally mounted
- Smooth turns are at constant speed so compensation should be easy

Without the Accelerometer, the observer still manages to provide a position estimate.

I believe it acts like a low pass filter when the accelerometer input is zero.

That is the circumstance when the mouse is off the ground or moving at constant velocity.

Feed forward is important

- Motor PWM throughout a profile needs large controller outputs for acceleration and speed.
- Without feed forward, the controller must provide these
- Controller can only give large outputs if there are large errors
- Feed forward provides the large outputs predictively
- Controller need only handle small errors.

Since the controller needs large errors to get the high PWM drive levels needed, some way must be found to provide most of the drive predictively.

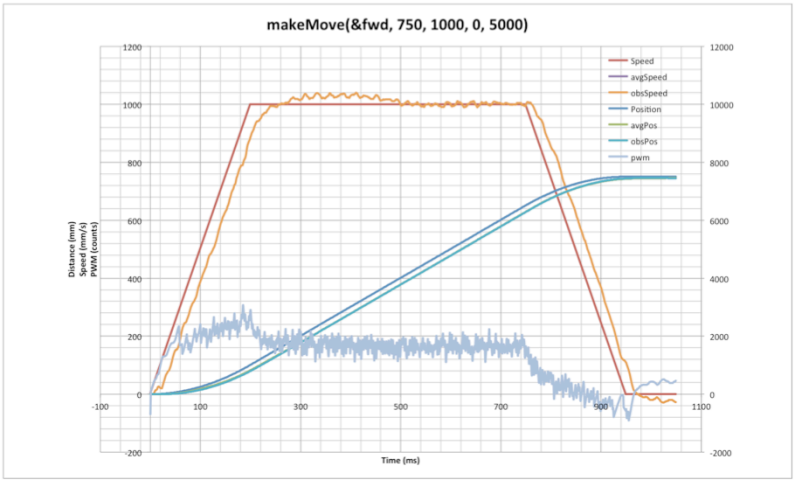
It is easy to see that a constant PWM would result in constant speed proportional to that drive.

Acceleration needs an additional boost to drive accelerating current through the motors.

There is a constant offset value needed to overcome losses and inefficiency in the drive.

If we can provide the right PWM values as a feedforward component, the controller should only have to make small adjustments

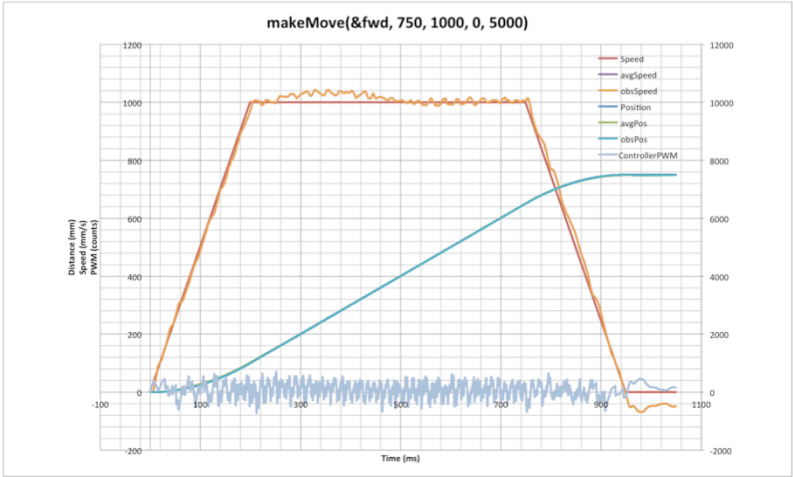
Tuned State Observer without Feed Forward



Without feed forward, it is clear that the error must grow before there is enough drive.

This creates lag in the response and will probably cause errors in the final position. and/or speed.

Tuned State Observer with Feed Forward



Minos 2018 Peter Harrison 26

By providing a suitable feed forward component, the robot will probably complete the move even with the controller disabled.

The controller is now only having to provide small amounts of corrective action as the encoders and estimator interact.

Evidence that the feed forward is correct is the observation that the average controller output is zero throughout the move.

(note that the behaviour at the end of the move is caused by not driving the wheels to zero long enough to overcome inertia)

Other Methods

- Kalman Filter for encoders plus IMU
- Used by Ng Beng Kiat
- Much more modelling and maths
- Not definitely better results

Minos 2018 Peter Harrison 27

The Kalman filter is very popular. It will adjust the gains as the system behaviour changes so that the resulting estimate is always as good as possible

It is also frequently misapplied and this may not even be the best place to use it.

Worse, it requires a greater understanding of the maths behind the process. It is not a simple case of plugging a few numbers into a couple of lines of code.

There is also some evidence that the Kalman filter will not produce any significant benefit over the State Observer approach:

J. H. Su, C. S. Lee and C. W. Chen, "Sensor fusion algorithms for encoder resolution enhancement in educational mobile robots," 2016 International Conference on Advanced Robotics and Intelligent Systems (ARIS), Taipei, 2016, pp. 1-5.

THANK YOU