# Building a competitive line follower

Mistakes made and lessons learned along the way

# Background

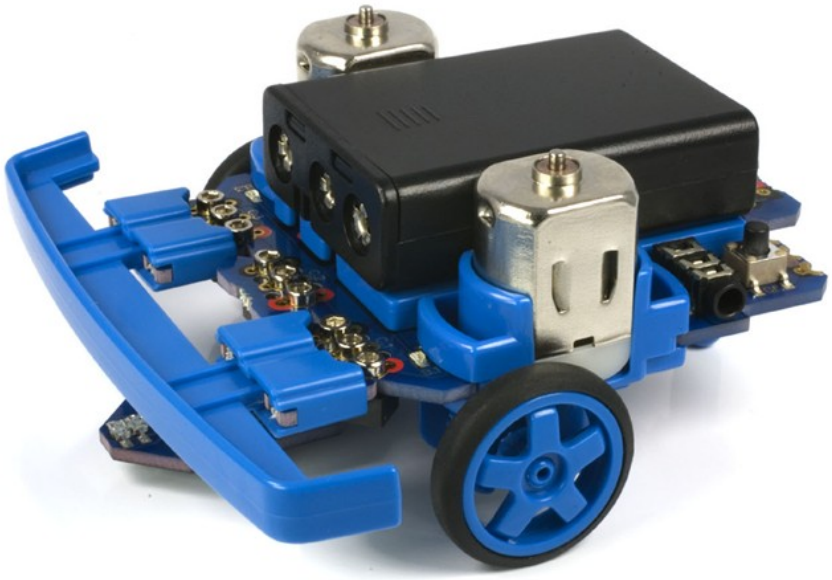
- Bought Lego Mindstorms for my son ~ 12 years ago
- Retired Xmas 2012 and started playing with robots
- Attended Techfest July 2013 as a spectator
- Tried to build a line follower for Robofest Nov 2013
  - Many mistakes and blind alleys
  - 2 weeks to go and no competitive robot
  - After lots of internet searching bought Pololu 3pi
  - After some re-programming ran 3pi in Robofest Nov 2013
  - Successful, but little satisfaction
  - However lots of lessons learned!
- Entered IET Robot Triathlon March 2014
  - Applied lessons learned earlier
  - Won competition, very satisfying!

# Mistakes and blind alleys

- Picaxe 20X2 Microbot

- Picaxe base and robot shield

# Picaxe 20X2 Microbot



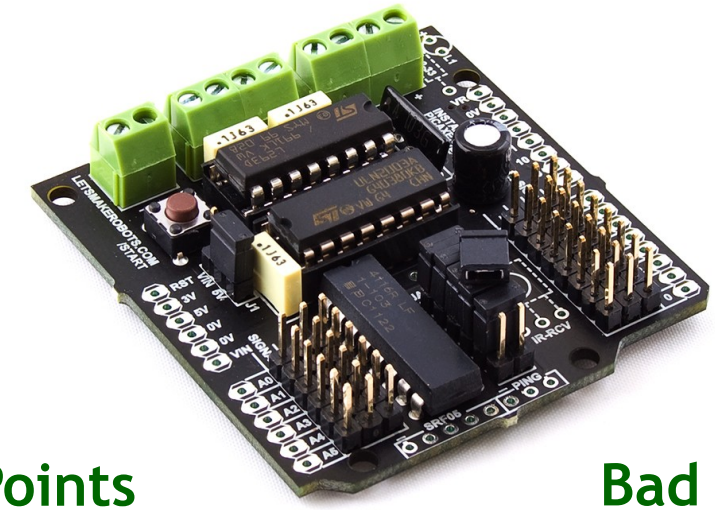
## Good Points

- Low cost
- Complete beginner’s solution
- Simple programming language
- Good range of sensors
- Great support

## Bad Points

- Poor drive train
- Slow and erratic
- Overall poor build quality

# Picaxe 28X2 shield base and robot shield




## Good Points

- Provides sensor i/o
- Provides motor drive
- Provides robot library
- Same Picaxe BASIC
- Software debug

## Bad Points

- Not cheap!
- Picaxe BASIC limited
  - No negative numbers
  - No floating point
- Difficult to implement PID control
- No motor braking on Robot Shield

# Not a blind alley!

- Pololu 3pi
- IET Robot Triathlon Winner

# Pololu 3pi

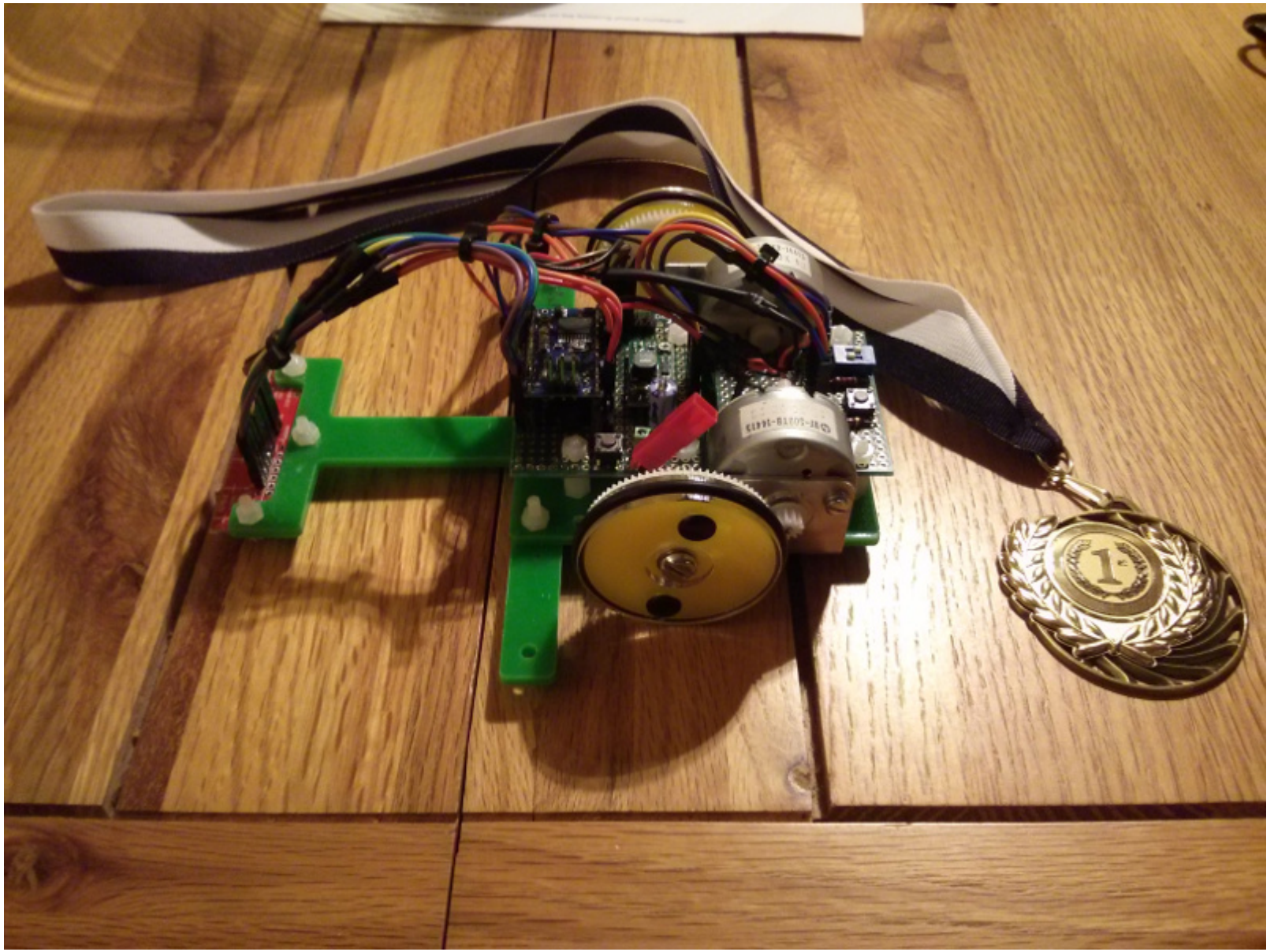

## Good Points

- Off-the-shelf line follower
- Good sensor array
- Power supply step up
- Fantastic robot library
- Great PID control

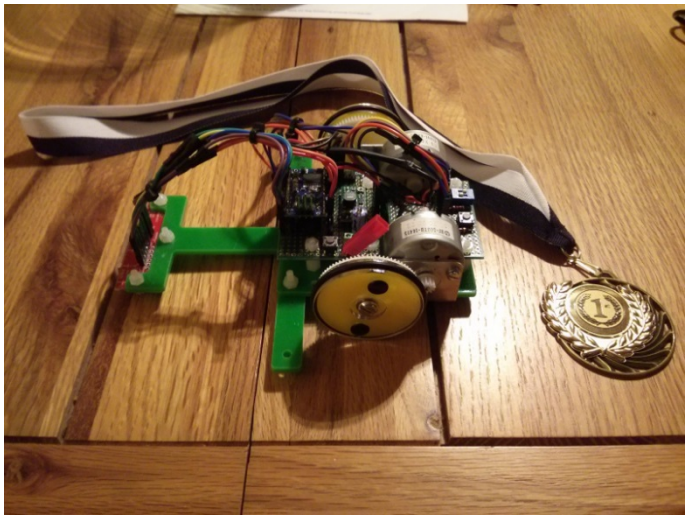
## Bad Points

- Needed to learn C in a hurry!

# IET Robot Triathlon Winner



# IET Robot Triathlon Winner



## Features

- Motor and drive train supplied
- Pololu sensor array
- Power supply step up
- Pololu Baby Orangutan controller
- Laser cut acrylic chassis

# Lessons Learned - General

- Robotics is a combination of mechanics, electronics and computing – don't overlook any aspect
- Don't re-invent the wheel! Lots of information and help is available on the internet
  - Micromouse Online
  - Manufacturer sites
  - Manufacturer libraries
  - User forums

# Lessons Learned - Mechanical

- Don’t skimp on quality of chassis, motors and drive train
- Go for a rigid and stable chassis
  - Custom laser-cut 3mm acrylic
- Build for minimum weight
- Fast turning requires a low moment of inertia, keep the weight near the COG
- Unexpected problems can have a mechanical cause!

# Lessons Learned - Electrical/Electronic

- Pololu have an excellent range of micro metal gearmotors

- Voltage step-up provides a high power stable motor voltage
  - Pololu modules

  - Cheap eBay modules
- LiPo batteries provide high current with low weight
  - Monitor LiPo voltage!

# Lessons Learned - PID control

- The key to line following is good PID control
- Proportional – Integral – Derivative
  1. Measure error e
  2. Correction = KP*e + KI * (integral e) + KD * (derivative e)
- Can be greatly simplified in our case:
  - Assume control loop time is roughly constant
  - Integral e = previous integral + new e (cumulative error)
  - Derivative e = new e – previous e (change in error)
  - Sign is very important!
  - Error is 0 when on line, - at one side of line and + at the other

# Lessons Learned - PID control

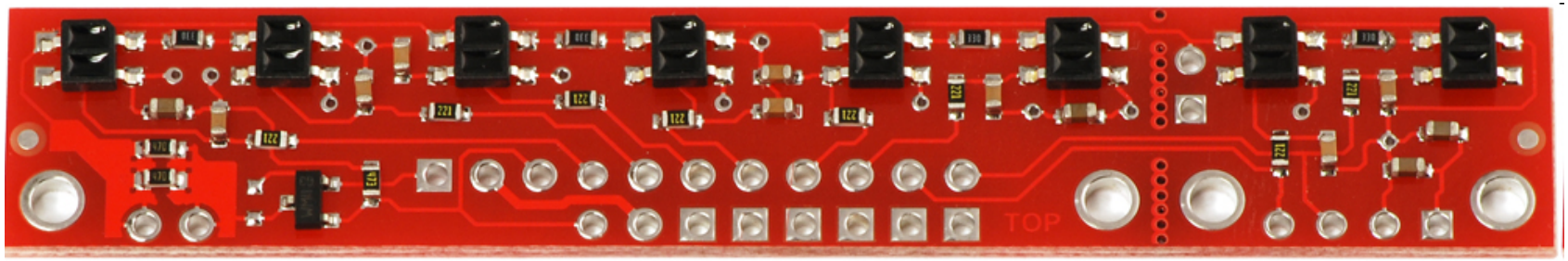
- The hard part is choosing good values for KP, KI and KD!
- Pololu 3pi line follower program is a good starting point:

  If each motor speed is a value between 0 and 255 with an upper limit of MAX, then a correction is applied by setting one motor to MAX and the other to (MAX-power_difference) where power_difference = (KP*prop + KD*deriv + KI*integ)

- This works on my robot with KP=0.06, KD=2.2, KI=0.0001
- Best values will depend on robot design
- Need to check bounds!

# Lessons Learned - Sensors

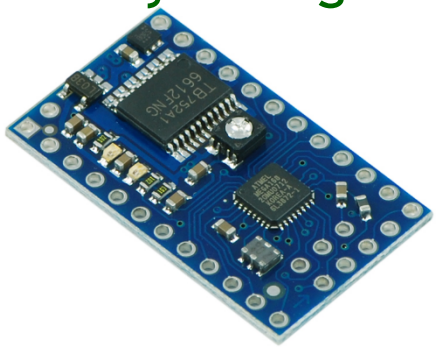
- Good PID control requires accurate error measurement
- I use an array of six ir sensors from Pololu




- Using Pololu library calibration routine gives error reading between -2500 and + 2500
- Sensors should not be too close to surface

# Lessons Learned - Controller

- I use Pololu Baby Orangutan



- AVR 328P microcontroller plus 2 x motor control channels
  - 1A continuous, 3A max
  - Input voltage 5V – 13.5V
  - 18 user i/o channels
  - 32KB flash memory, 2KB RAM
  - Extensive robot library and support
  - Program in C or C++

# Still to do

- Proper debug system
- Data logging
- Use curvature markers
- Map track using accelerometer?