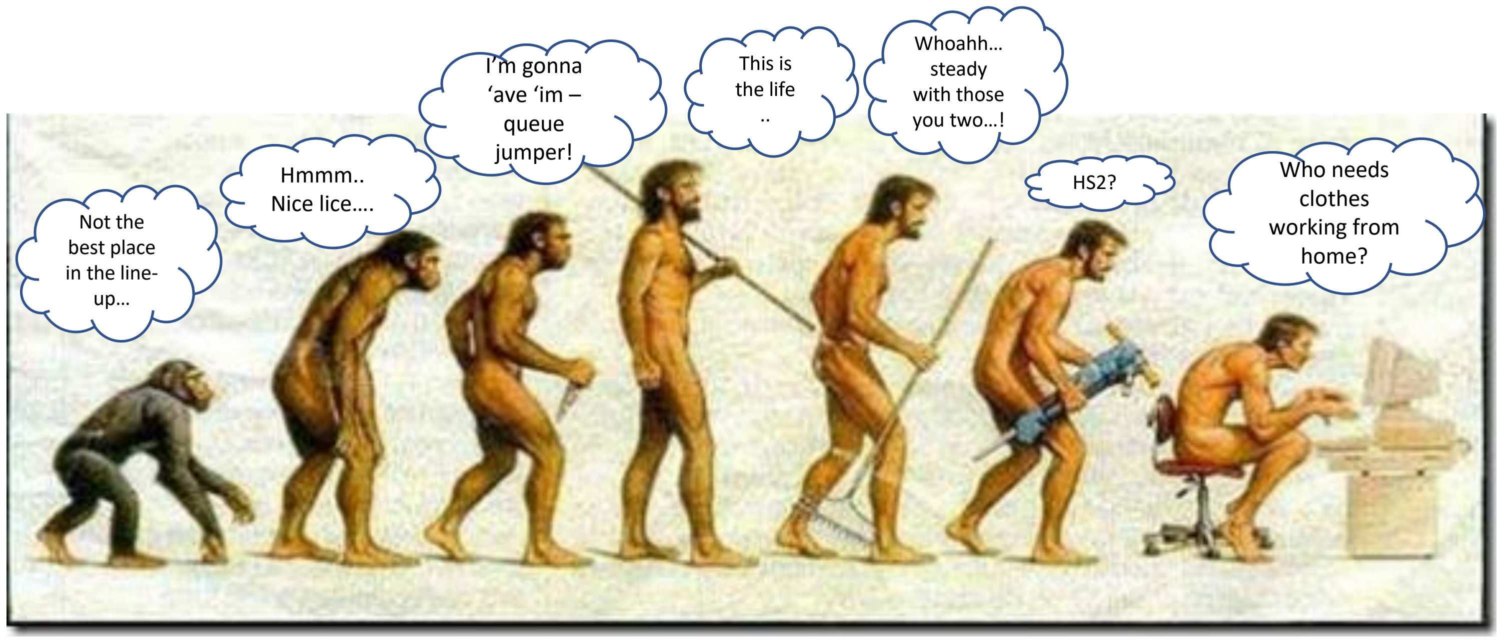
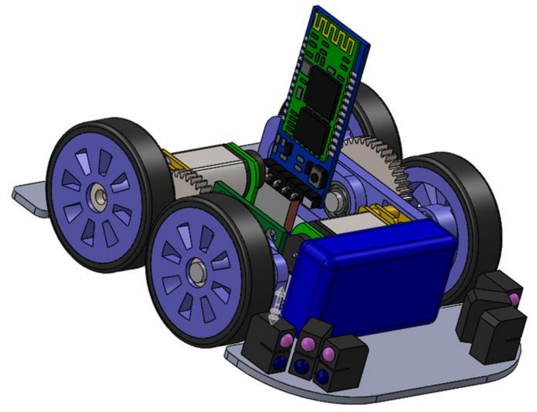
Designing a Teaching Mouse ? – Rather …

“Evolution of a Mouse for Teaching Embedded Systems and Control” by Ivan Eumowz


Not the best place in the line-up…
Hmmm.. Nice lice….
I’m gonna ‘ave ‘im – queue jumper!
This is the life ..
Whoahh… steady with those you two…!
HS2?
Who needs clothes working from home?



“The ascent of MENG”

Designing a mouse is something you have all done (or will do) …

… when “Teaching” is added, each of us will have a very different idea of what is required.

My focus for this presentation is on teaching Embedded Systems with a Micromouse.

It is often said that we should learn from history ……

Please bear with me while I rediscover what I can remember of my history with micromouse!

Historical musings

I used to teach microprocessors – Z80, 8051
external address/data bus, ROM, RAM and I/O timing calculations (datasheets!),
8255 PPI, LS138 decoders, discrete logic …..

Then the 87196CA – an 8/16bit automotive microcontroller with external address/data bus and CAN – using a custom board running MPE FORTH. More timing diagrams and datasheets.

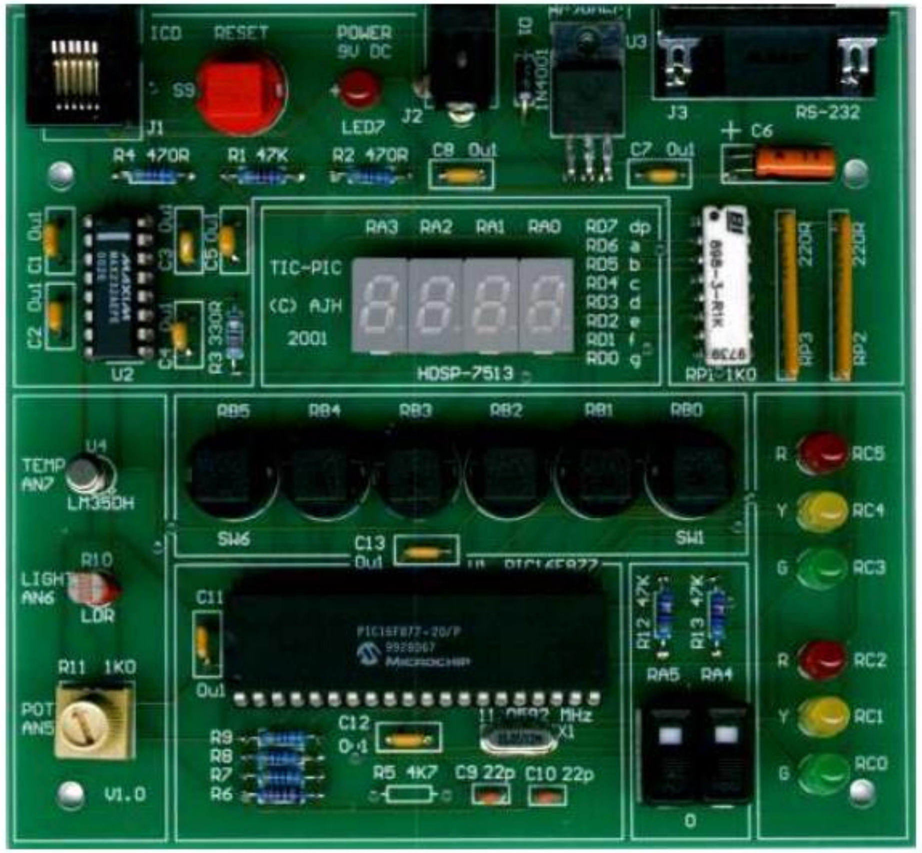
At one point we taught all three processors in the same module – times change.

TICPIC came along (esteemed colleague Andrew Hill) – used for second year embedded systems from late 90's to present (Microchip 16F877A to 18F4520). Timing calculations/decoder design forgotten – but who needs timing calculations and decoder design with a microcontroller?

Merging of the 2nd and 3rd year teaching came with using the PIC16F877A for both *TIC-PIC and RoboTIC *

**A lot of TIC here ….. The "not-to-be-used" abbreviation for the Technology Innovation Centre.*

*TIC-PIC v1.0 AJ Hill - 2001



(You can see how driven we were to include ‘TIC‘ in everything!)

This board was designed to be a stand-alone target system, Effectively unbreakable as there were no external connections.

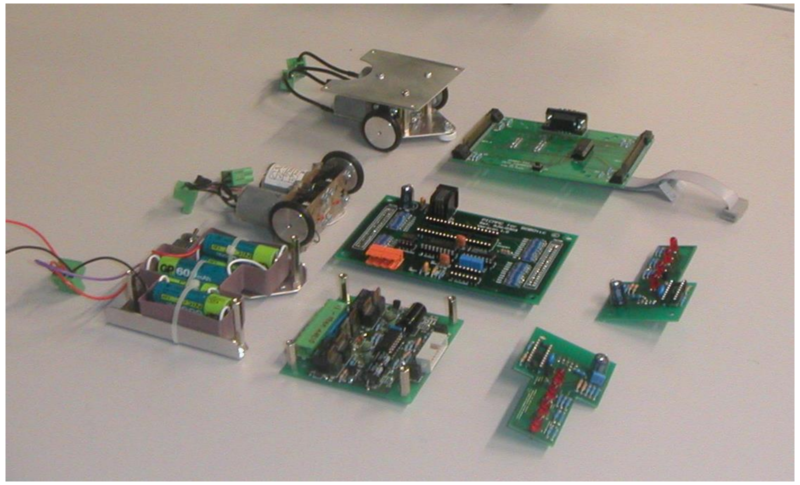
Excellent for introductory embedded systems – PIC16F877

In later years two further revisions were developed:

1. Two buffered I/O ports (MatrixMultimedia compatible) were added to extend its capability – *TICPIC2.

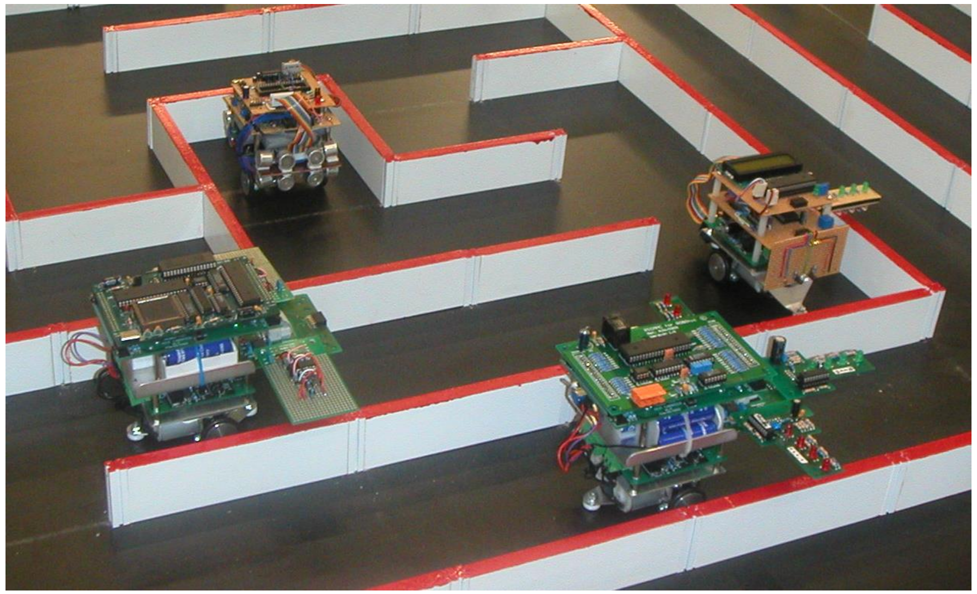
2. A 75mm square SMD version of TICPIC1 with on-board PicKit2 debugger and USB serial comms was developed – *TICPIC3
Plug a USB lead into a laptop and it was up and running. The aim was to give these to students for home use.

BEng Electronic Engineering – 3rd Year Embedded Systems – RoboTIC*

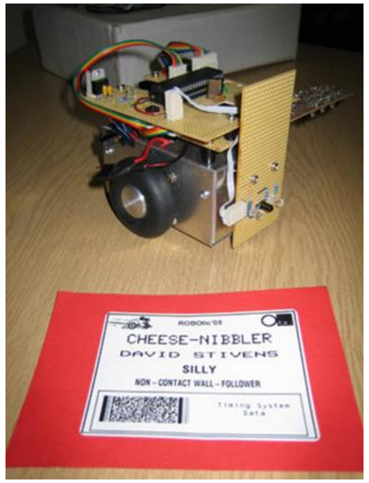


Modules developed gradually - 2002 – 2006. Note the use of Swallow Systems motor assemblies.

RoboTIC modules (left)
3rd year projects (right)

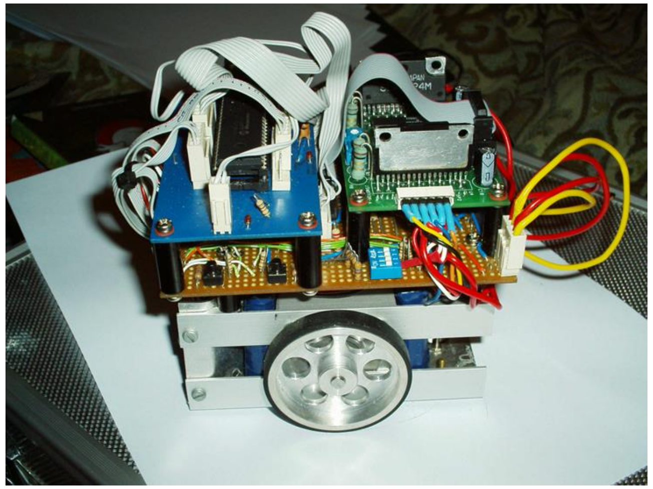


Initially FORTH on 87C196CA processor platform developed as a result of commercial activities.
Later PIC16F74, PIC16F877A.
Swallow Systems drive train, L298 motor drivers, ‘intelligent’ sensors giving distributed processing.



Later projects:

Dave Stivens (left),
Richard Nock (right)
Both competed in the UKMM2006 with their final year project mice.



Told you so ….

RoboTIC* … Lessons learned, problems identified, time to move on …

Advantages

- Modular construction, distributed processing
- Good for design and build exercises

Disadvantages

- Mechanical problems with the drive train – gear stripping
- High current motor (3.5A stall)
- Limited sensitivity from the ‘wing’ sensors
- Limited control with the chopper/relay drive system
- Limited resolution from the encoders
- Expensive to build, time-consuming to maintain.
- Inelegant and slow

**And there it is again …..*

Studmouse* - 2007/8 - A staff/student development project (Blanca and Cristina from UPM)

New Teaching platform – 18F4520 processor

Symmetrical front/rear IR and Ultrasonic sensors

Spur and Pinion gearing – no more stripped gears.

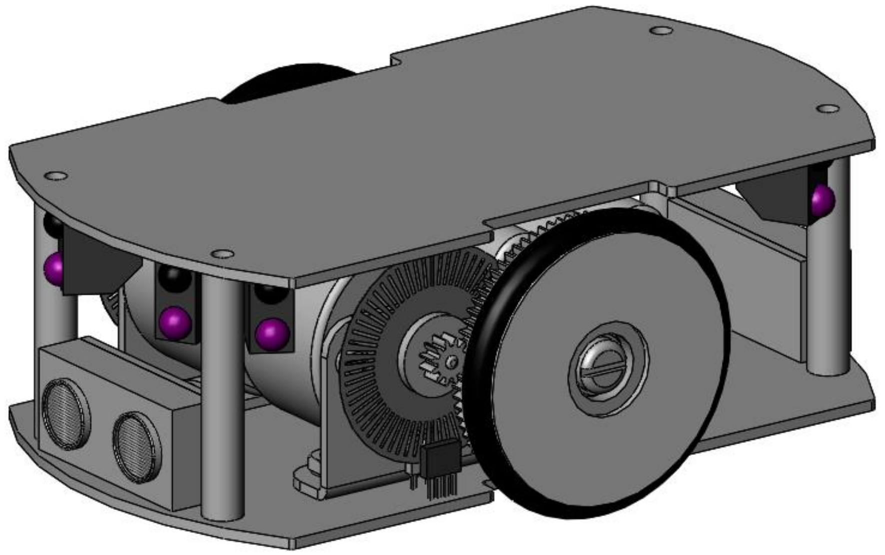
Low-cost/ low-power motors

Alternate comms bus (I2C) for additional sensors (e.g. US)

Up to 6* encoder resolution ➔ 800ppr

RF and wired RS232 comms for better run-time debugging

Lower cost and better performance cf. ROBOtic

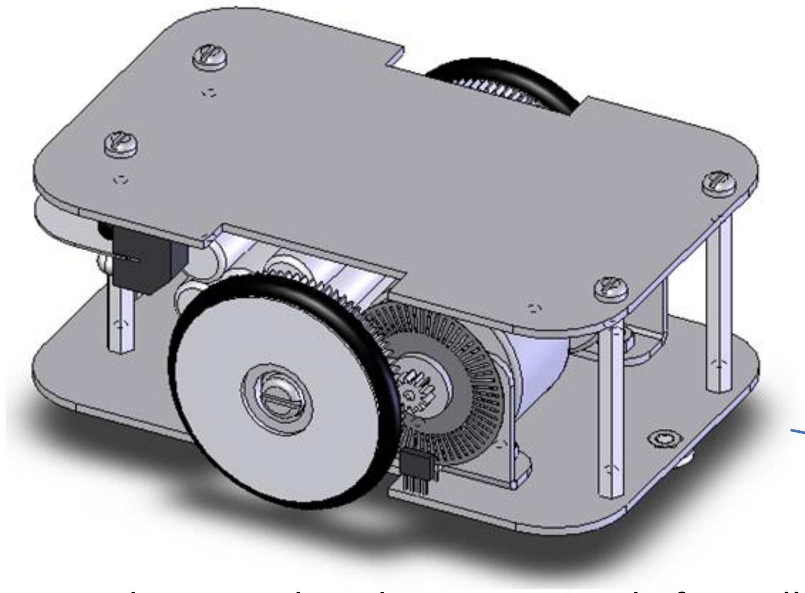


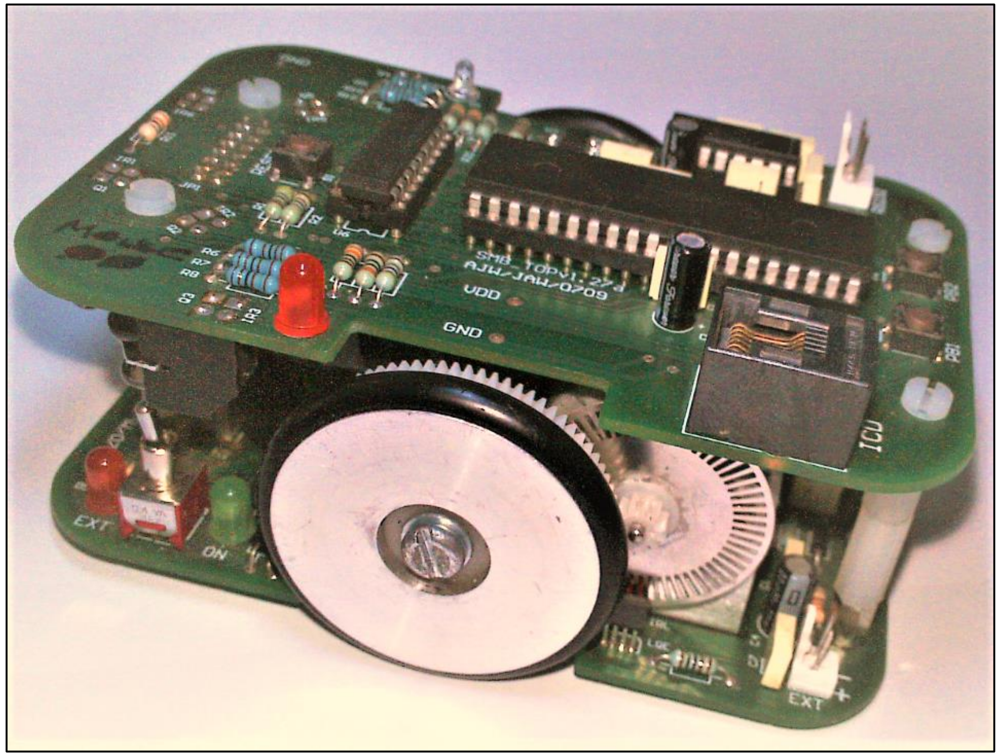
Thought to be a little too much for teaching so simplified…

(SMA => SMB => MyTEEmouse) but retained the drive train.

**Originally called AutomaTIC – can't get away from it!*

MyTEEmouse* – developed as a teaching platform for Embedded Systems and Control





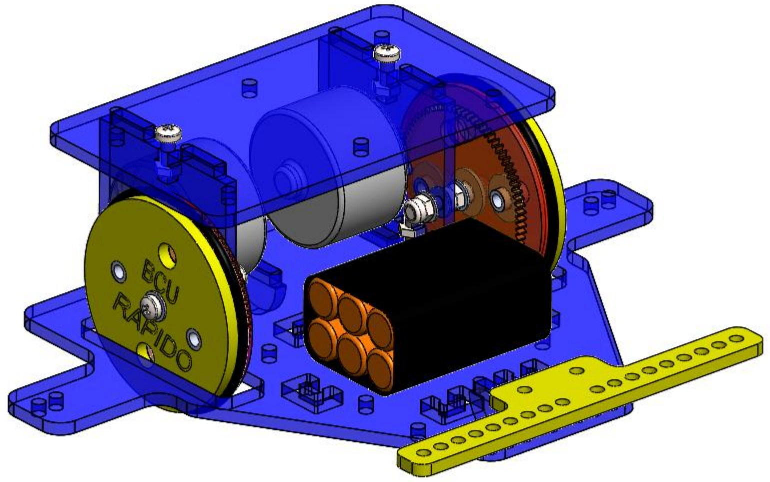
Students produced own main code for wall-follower from a set of base libraries developed throughout the module.
18f4520 processor, LM293D motor driver, RS232 comms, PICkit2 debug, SPI FRAM, 6 wall sensors, + sufficient I/O for sensors and encoder. (encoders optional extra for project students).

No build requirement but this was moved down to second year. Perfectly feasible to build top or bottom board – even both - as student project. No-one did.

MyTEEmouse. Developed 2008/9 was the teaching platform for 3rd year Embedded Systems and Control from 2010 to 2017. Re-introduced 2021. A success story.

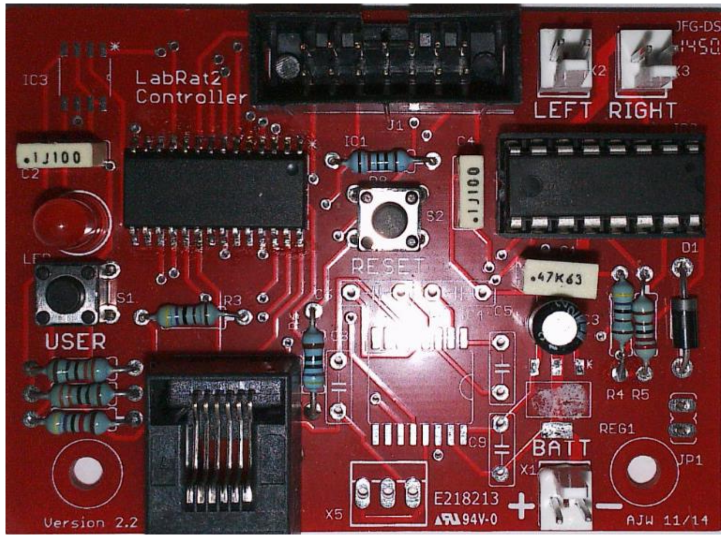
** As we were by this time the Faculty of Technology, Engineering and the Environment (TEE)*

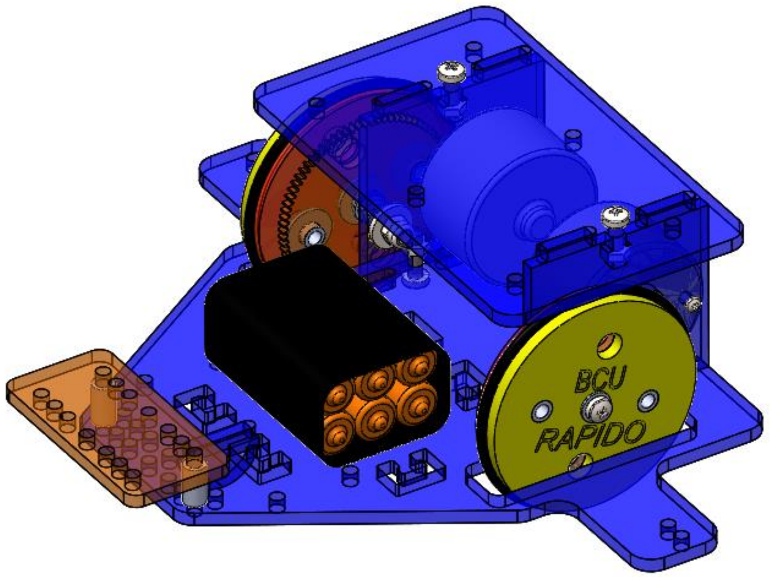
BEng Electronic Engineering 2nd Year Electronics Project 2012 : LabRat – a Line Follower



Students were given a pcb to build – mixed thru-hole and surface-mount devices. This was usable as a stand-alone target board.
They produced their own 'main' code from a set of base libraries.

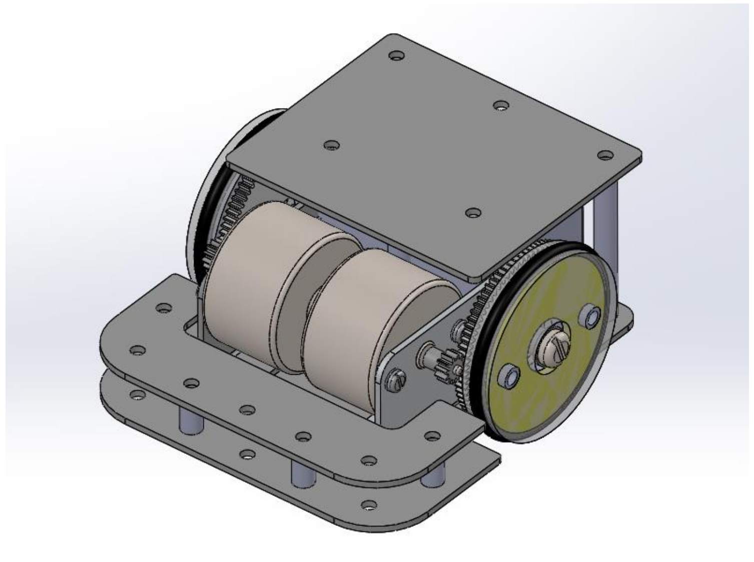
18f2520 processor, LM293D motor driver, RS232 comms, PICkit2 debug, SPI FRAM + sufficient I/O for sensors.
i.e. a subset of MyTeemouse hardware, and the same processor and tool-suite as the TICPIC static platform.

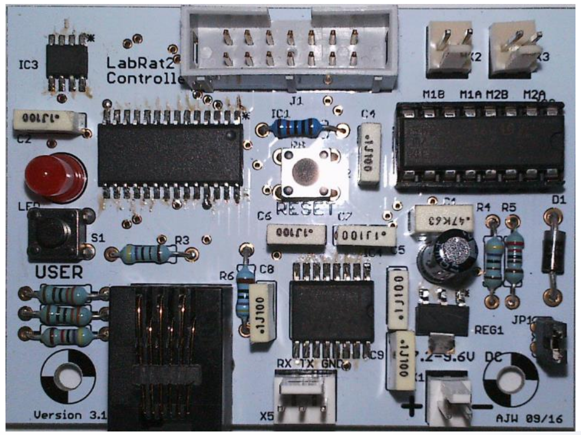




** Originally called Rapido, as Rapid Electronics were very interested in selling it – Don't ask why it didn't happen ......*

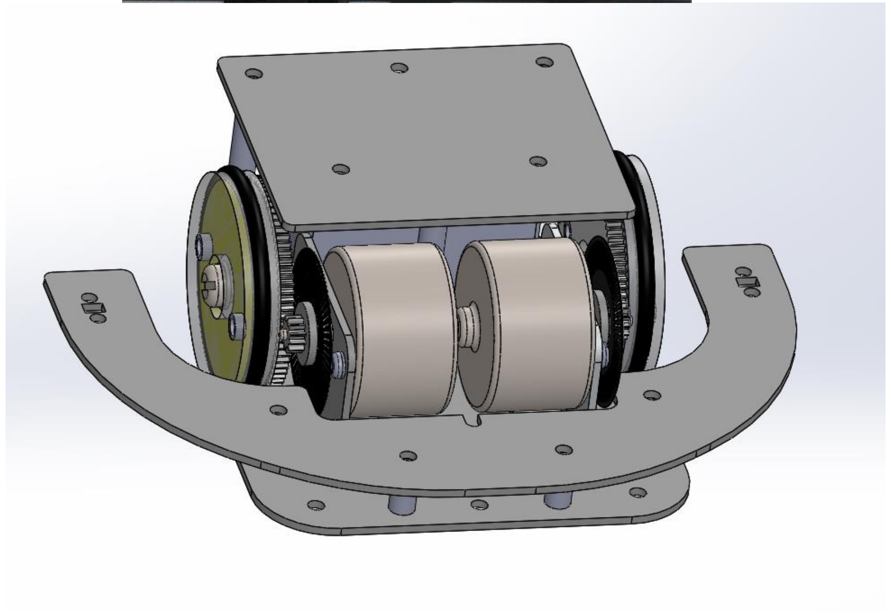
IET Robot Triathlon 2014 and BEng Electronic Engineering 2nd Year Embedded Systems





First batch of IET mice used Myteemouse motor mounts and gearing as we had stocks. Chassis, wheels and 80-tooth spur gear was laser-cut from 3mm Perspex.

Labrat controller board was used with these bases in Second Year Embedded Systems teaching, replacing the single-function Line Follower platform. Students were given a choice of LF/WF.



Mechanical base configurable as either a Wall Follower or Line Follower

At this point we had created a family of compatible devices for each level of the course, and we had a course that had all requisite elements for successful project work.

Myteemouse
Very successful for students, staff development and Open Days for parents and prospective students.

LabRat was in continual development from 2012 through to 2016 eventually using N20 motors.

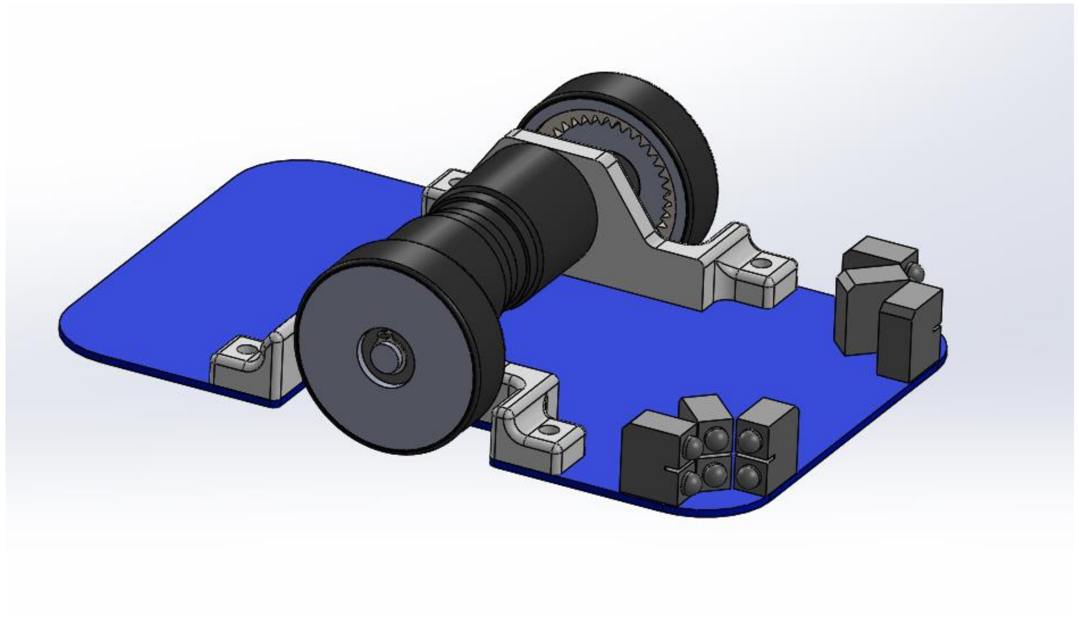
A key successful element for 2nd year teaching was the **LabRat controller board** itself.

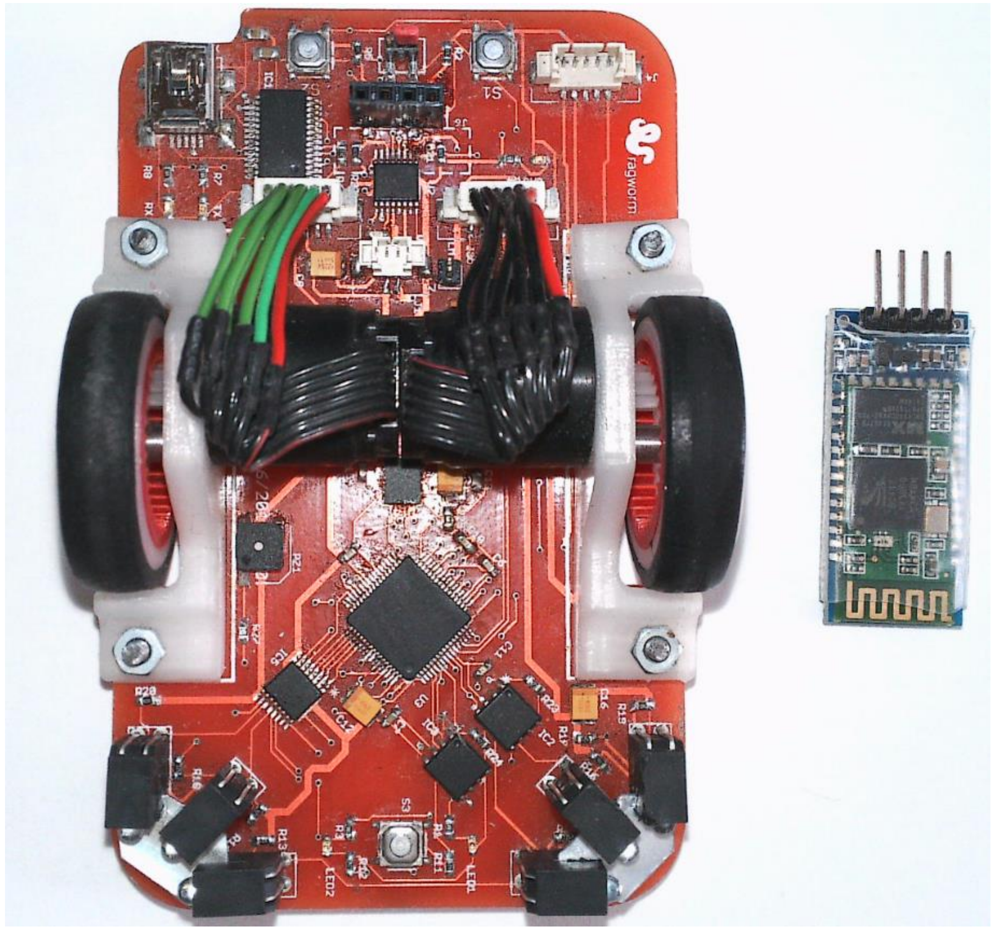
- the students built and debugged it themselves
- Sufficient I/O for Line and Wall follower, and of course could be used for many other things
- Sufficient I/O for optional encoders
- Code and peripheral compatibility with TICPIC and a logical next step
- It was NOT a structural component of the robot design giving flexibility in use
- It was a stand-alone PIC18F2520 target system with on-board H-Bridge.

TICPIC 1,2 used mainly in 2nd year teaching for introduction to Embedded Systems and C programming.

We also had **a first year module "Engineering Practice"**, that taught Solidworks and Eagle PCB on a project led programme involving design and manufacture of a finished product. Students designed a PCB, had it manufactured in-house, and designed an enclosure for 3D print.

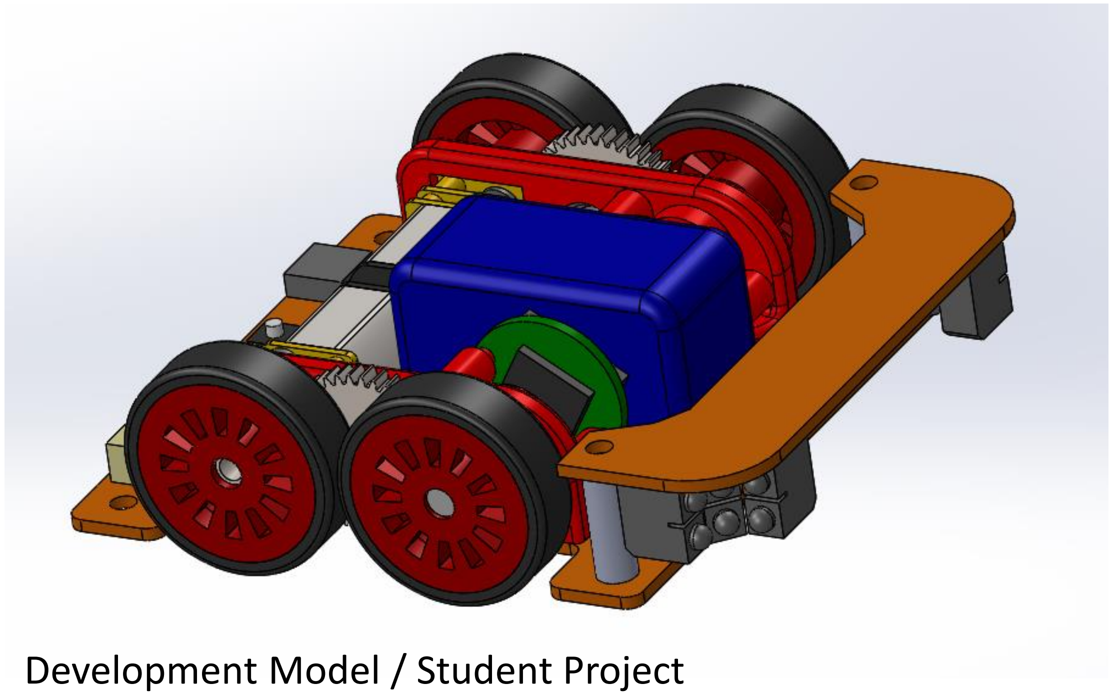
A bit of ‘me’ time …. A step up to the DSPIC33EP and MPLABX MiniMouse/ PUMBA - 2014/15



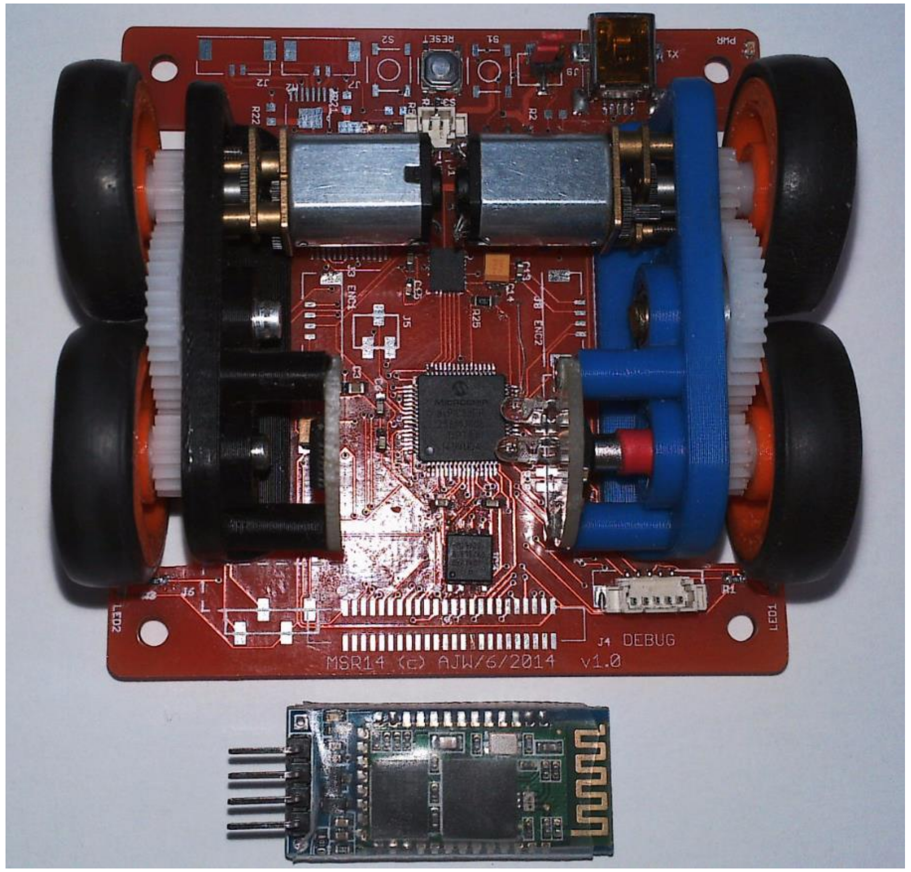


Personal development model
Faulhaber Motors
9:36 pinion + ring gear for minimum width (65mm x 100mm)
16-bit DSPIC33EP256MU806 processor
Gyro/Accelerometer – LSM330DLC
FRAM, Bluetooth and USB comms for data acquisition
Nylon 3D printed wheels and motor mounts..

3rd Year BEng Electronic Engineering Project Placement Student from UPM - MSRV2 - 2014

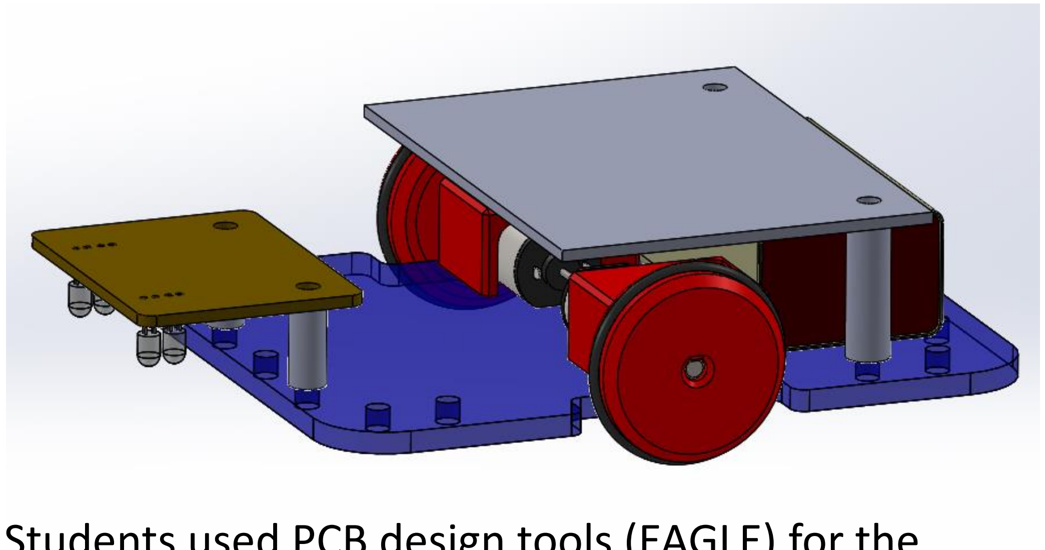


Development Model / Student Project
(Manuel Serrano Revuelta => MSR)
Mirrored 10:1 N20 Motors
AS5045 magnetic encoders (in-house) to reduce width (75mm x 75mm)
16-bit DSPIC33EP256MU806 processor
Designed as a base for both line and wall sensors

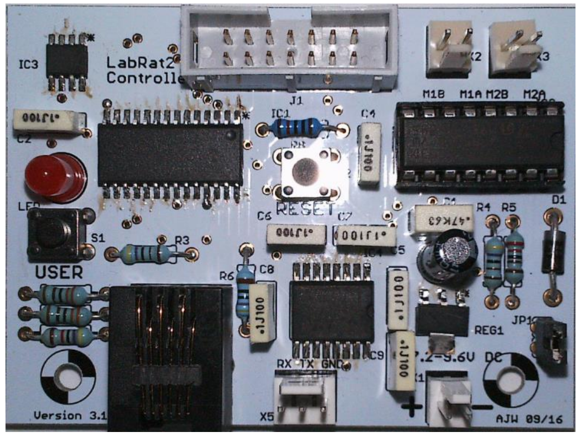


This is where the 4-wheel N20 drive originated to give a slave axle for the magnetic encoders.

BEng Electronic Engineering 2nd Year Electronics Project – N20 motors - YALF 2016

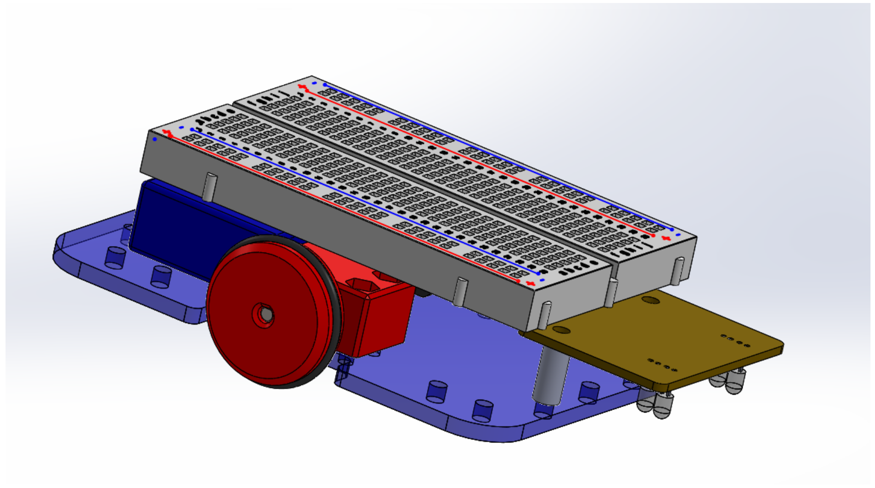


LabRat 2016 controller board, self-build sensor board



Students used PCB design tools (EAGLE) for the sensor board with in-house manufacture. They also built and tested the LabRat controller board. A base set of libraries was provided for test and teaching purposes.

N20 motors used for first time for a teaching platform. Base laser-cut, wheels and motors 3D printed.



Configured for breadboard/module build.

YALF 2016

2017 – 2021: The Dark Ages …..

In July 2017 line-follower and wall-follower were a focus for design, build and programming at all years for Electronic Project, Embedded Systems, and Embedded Systems and Control.

All students that reached the final year had experience of PCB and Mechanical design, and were relatively adept at C programming using MPLAB and the PIC18F4520 family.

By September 2021 -

- MyTEEmouse and LabRat no longer used – new teaching staff, different approaches, course changes.
- Inclusion of Arduino in teaching at 2nd year 1st semester, Microchip TICPIC at 2nd year 2nd semester
- UKMARSbot used in 2nd Year
- Use of TICPIC for teaching 3rd year Embedded Systems and Control
- No 3D modelling, no PCB design, no build experience
- No lab-time as a result of COVID19.

In 2021/22
With no suitable alternative available for Embedded Systems and Control MyTEEmouse was resurrected and used to teach an appropriate syllabus based on the development of a DC-motor PD positional controller. I also picked up a 4th year Project Student!

MEng 4th year Project 2021/22: Geraint Leahy and Anthony Wilcox

“The design and development of a small autonomous robot for teaching robotics to engineering students with the aim of entering the UK Micromouse competition”

Initial assessment

1. Mechanics/Motors : Possibly the biggest issue for potential builders
    The only suitable cheap COTS drive-train is the N20 motor assembly with optional crude encoders.
    Other motors require custom mounts, gearing and encoders.
    Wheels and tyres – not easy to find. 3D print what we can.

2. Power:
    Lipo, NiCd, PP3 alkaline? – PP3 for teaching.

3. Program development
    Development tools : IDE, Compiler - assume C/C++?
    Debuggers/Simulators/Target boards

4. Processor choice - based on target user, simplicity or capability?
    Single option or multi choice? - disadvantage of multi-choice is additional workload for resource generation
    PIC8,16,32 bit ;   ATMEL/Arduino;   STM32;   ESP-32; Other?

5. Communications: wired/wireless/both

6. On-board peripherals: FRAM, Gyro/Accelerometer. Use of modules where appropriate.

There are more questions than answers ….

Who are the target users? - University / College / School / Club / Individual?
What are the target disciplines? - Software Engineering (SE), Electronic Engineering(EE), Mechatronics(ME), Control Engineering (CE), Mechanical (M)
What is the teaching element – hardware design, software, control, fabrication, all of the above?

What are the requirements for a micromouse teaching platform?
Should it be a kit? Should it be a 'build' or a use 'platform'?
What level of documentation is required?
How much software should be provided? - base functionality, full demo code?
Should a virtual teaching tool be used?
How much should it cost? What cost development tools?

Is a micromouse/line-follower the best option?
Would single-function platforms be a better option?

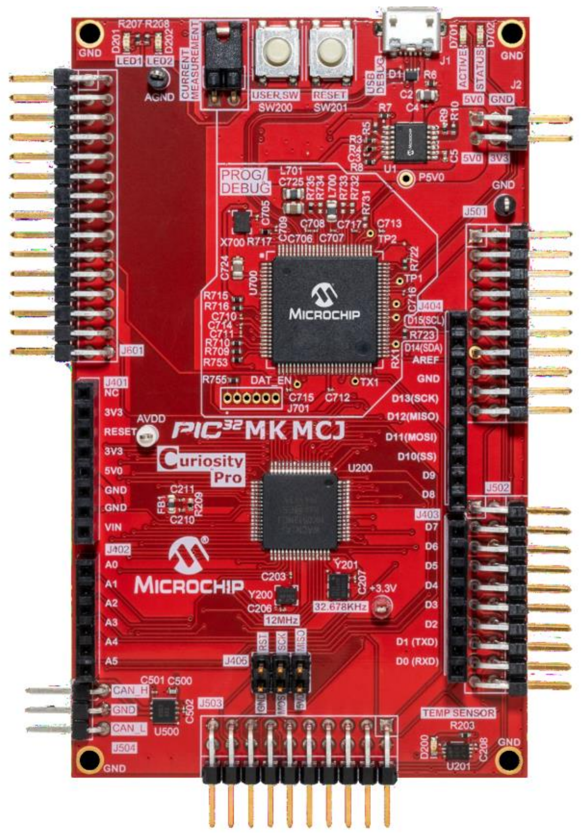
Does it need to be competitive? – if so at what level?
What processors to use? 8/16/32 bit – does it matter?
What language to use?
IDE – should we follow the trend for code generators?

…………and many, many more …..

Far too many questions with different answers from everyone I suspect …

So I set my target users as Final year Degree in any Engineering discipline, and I chose:

- **Microchip** as I was familiar with MPLABX for 8/16 bit devices, and the tool-suite is free.
- A **32-bit** motor control microcontroller with Hardware **FPU** and available **target board**
- A **4-wheel mouse** as I already had a workable drive-train (2014) that could be modified and 3D printed.
- Extended shaft **N20 motors with magnetic encoders** as they are cheap and available.
- **90mm long by 70mm wide**, based on a reasonable minimum spacing of the interleaved drive trains.
- A sensor configuration I had used before that worked well.
- A user I/O configuration I had used before that worked well.
- A mix of PTH and SM devices as it was a real pain to build the previous one.
- **Modules** for motor driver, gyro and Bluetooth – code development in advance and less SMD soldering.
- COST? No point in worrying – Motors as cheap as chips and everything else is similar for any design.



PIC32MK MCJ Curiosity Pro Features

- PIC32MK0512MCJ064, 120 MHz, 512 KB Flash, 64 KB SRAM
- On-Board debugger (PKoB4)
  - Real time Programming and Debugging
  - Virtual COM port (VCOM)
  - Data Gateway Interface (DGI)
- Arduino Uno R3 compatible interface
- Xplained pro extension compatible interface
- Motor Control interface
- On-Board Temperature Sensor
- CAN interface
- User buttons
- User LEDs

The PIC32MK0512MCJ064 processor has everything a micromouse desires, and more….

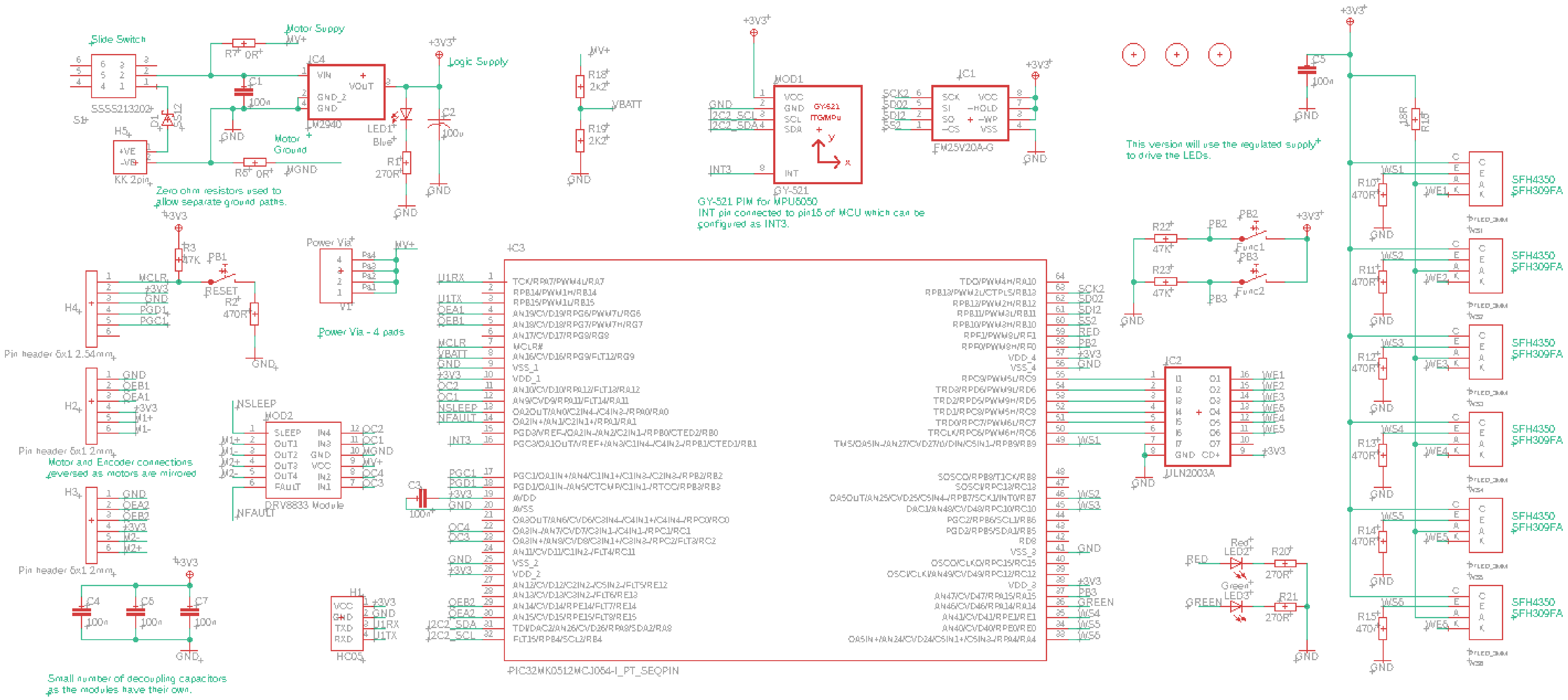
Multiple PWM options, nine Output compare modules, three Quadrature Encoder Interfaces, multiple 16/32-bit timers, two I2C and two SPI modules, up to 30 analog inputs to 8-12bit ADC, GPIO and Programmable Pin Select on most pins.

Meng Electronic Engineering Group Project – 4th Year Student + staff support

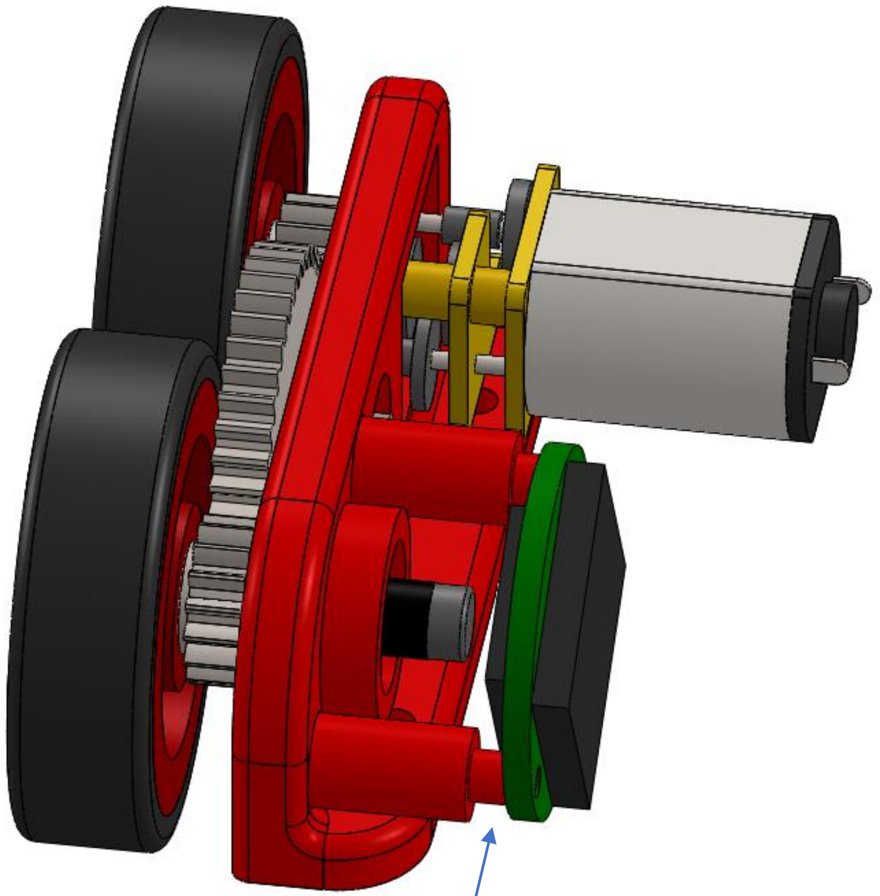
Pin assessment for processor

No of pins	Function	Type	
3	Debug and Reset	Digital	PGC,PGD,MCLR
2	RS232	Digital	TX,RX
2	I2C	Digital	SCL, SDA
3+2	SPI	Digital	SCK,SDO,SDI,CS1,CS2
4	Encoders	Digital	CH1A,CH1B,CH2A,CH2B
6	Detectors	Analogue	PT1-6
6	Emitters	Digital	IR1-6
1	Battery Monitor	Analogue	VBATT
4	User I/O	Digital	PB1,PB2,LED1,LED2
10 (!)	Power Supply	Power	VDD x 4, VSS x 4, AVDD, AVSS
6	Motors	Digital	NFAULT,NSLEEP,PWM1A,PWM1B,PWM2A,PWM2B

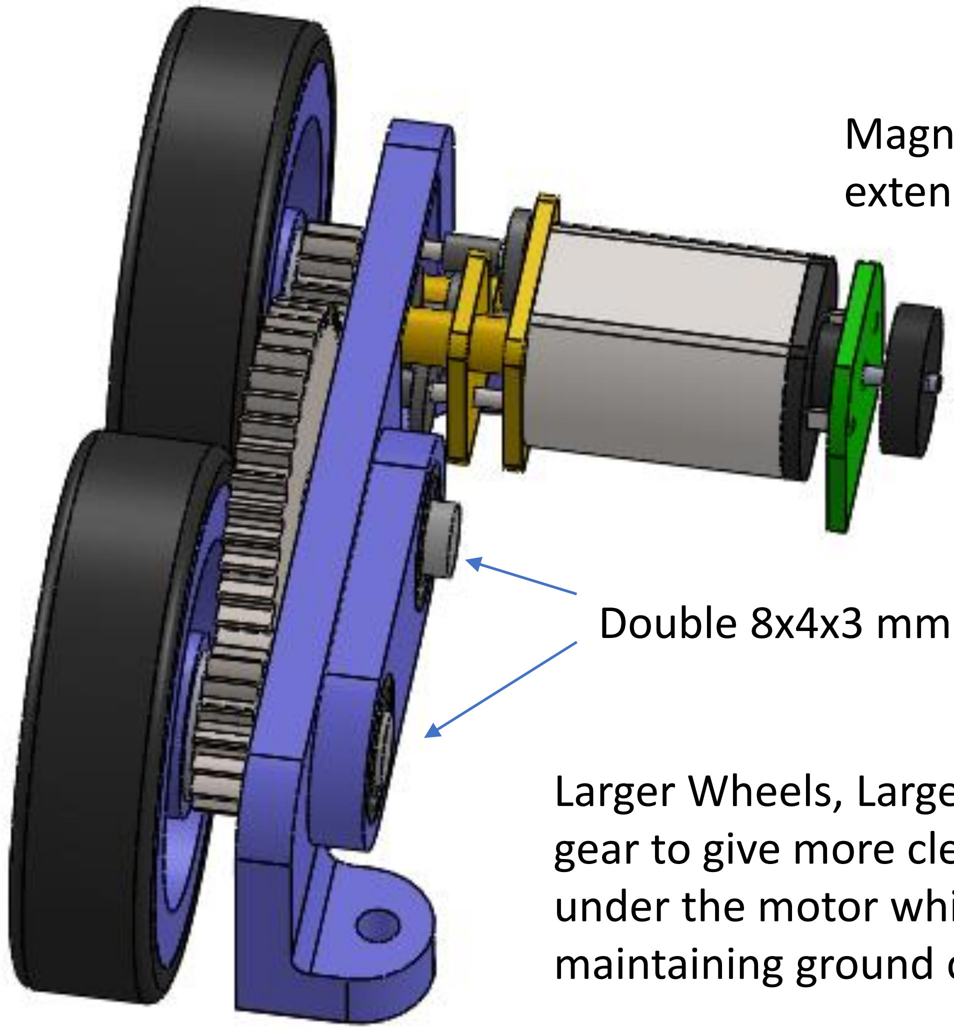
49 pins total – so I used the 64-pin version, added flexibility with PPS pin placement.


Slide Switch
S5S5213202
S1
Motor Suppy
MV+
R7 0R
IC4
VIN
VOUT
GND_2
GND
M2940
C1 100n
GND
+3V3
Logic Supply
LED1 Blue
R1 270R
C2 100u
H5
+VE
-VE
KK 2pin
R5 0R
MGND
Motor Ground
Zero ohm resistors used to allow separate ground paths.
MV+
R18 2K2
VBATT
R19 2K2
GND
MOD1
VCC
GND
SCL
SDA
INT
GY-521
ITG/MPU
I2C2_SCL
I2C2_SDA
INT3
GY-521 PIM for MPU6050
INT pin connected to pin16 of MCU which can be configured as INT3.
IC1
SCK
SI
SO
-CS
VCC
-HOLD
-WP
VSS
SCK2
SDO2
SDI2
SS2
FM25V20A-G
C5 100n
R8 18R
This version will use the regulated supply to drive the LEDs.
SFH4350
SFH309FA
WS1
WE1
WS2
WE2
WS3
WE3
WS4
WE4
WS5
WE5
WS6
WE6
R10 470R
R11 470R
R12 470R
R13 470R
R14 470R
R15 470R
R3 47K
PB1
RESET
R2 470R
H4
MCLR
+3V3
GND
PGD1
PGC1
Pin header 6x1 2.54mm
Power Via
V1
Pa1
Pa2
Pa3
Pa4
Power Via - 4 pads
IC3
PIC32MK0512MCJ064-I_PT_SEQPIN
U1RX
U1TX
QEA1
QEB1
VBATT
OC2
OC1
NSLEEP
NFAULT
C3 100n
OC4
OC3
QEB2
QEA2
I2C2_SDA
I2C2_SCL
TCK/RPA7/PWM4L/RA7
RPB14/PWM1H/RB14
RPB15/PWM1L/RB15
AN19/CVD19/RPG6/PWM7L/RG6
AN18/CVD18/RPG7/PWM7H/RG7
AN17/CVD17/RPG8/RG8
MCLR#
AN16/CVD16/RPG9/FLT12/RG9
VSS_1
VDD_1
AN10/CVD10/RPA12/FLT13/RA12
AN9/CVD9/RPA11/FLT14/RA11
OA2OUT/AN0/C2IN4-/C4IN3-/RPA0/RA0
OA2IN+/AN1/C2IN1+/RPA1/RA1
PGD3/VREF-/OA2IN-/AN2/C2IN1-/RPB0/CTED2/RB0
PGC3/OA1OUT/VREF+/AN3/C1IN4-/C4IN3-/RPB1/CTED1/RB1
PGC1/OA1IN+/AN4/C1IN1+/C1IN3-/C2IN3-/RPB2/RB2
PGD1/OA1IN-/AN5/CTCMP/C1IN1-/RTCC/RPB3/RB3
AVDD
AVSS
OA3OUT/AN6/CVD6/C3IN4-/C4IN1+/C4IN4-/RPC0/RC0
OA3IN-/AN7/CVD7/C3IN1-/C4IN1-/RPC1/RC1
OA3IN+/AN8/CVD8/C3IN1+/C3IN3-/RPC2/FLT3/RC2
AN11/CVD11/C1IN2-/FLT4/RC11
VSS_2
VDD_2
AN12/CVD12/C2IN2-/C5IN2-/FLT5/RE12
AN13/CVD13/C3IN2-/FLT6/RE13
AN14/CVD14/RPE14/FLT7/RE14
AN15/CVD15/RPE15/FLT8/RE15
TDI/DAC2/AN26/CVD26/RPA8/SDA2/RA8
FLT15/RPB4/SCL2/RB4
TDO/PWM4H/RA10
RPB13/PWM2L/CTPLS/RB13
RPB12/PWM2H/RB12
RPB11/PWM3L/RB11
RPB10/PWM3H/RB10
RPF1/PWM8L/RF1
RPF0/PWM8H/RF0
VDD_4
VSS_4
RPC9/PWM5L/RC9
TRD3/RPD6/PWM9L/RD6
TRD2/RPD5/PWM9H/RD5
TRD1/RPC8/PWM5H/RC8
TRD0/RPC7/PWM6L/RC7
TRCLK/RPC6/PWM6H/RC6
TMS/OA5IN-/AN27/CVD27/LVDIN/C5IN1-/RPB9/RB9
SOSCO/RPB8/T1CK/RB8
SOSCI/RPC13/RC13
OA5OUT/AN25/CVD25/C5IN4-/RPB7/SCK1/INT0/RB7
DAC1/AN48/CVD48/RPC10/RC10
PGC2/RPB6/SCL1/RB6
PGD2/RPB5/SDA1/RB5
RD8
VSS_3
OSCO/CLKO/RPC15/RC15
OSCI/CLKI/AN49/CVD49/RPC12/RC12
VDD_3
AN47/CVD47/RPA15/RA15
AN46/CVD46/RPA14/RA14
AN41/CVD41/RPE1/RE1
AN40/CVD40/RPE0/RE0
OA5IN+/AN24/CVD24/C5IN1+/C5IN3-/RPA4/RA4
RED
PB2
PB3
GREEN
R22 47K
R23 47K
Func1
Func2
IC2
ULN2003A
I1
I2
I3
I4
I5
I6
I7
O1
O2
O3
O4
O5
O6
O7
CD+
Red LED2
Green LED3
R20 270R
R21 270R
H2
Pin header 6x1 2mm
M1+
M1-
Motor and Encoder connections reversed as motors are mirrored
NSLEEP
NFAULT
MOD2
SLEEP
OUT1
OUT2
OUT3
OUT4
FAULT
IN4
IN3
IN2
IN1
DRV8833 Module
M2+
M2-
H3
QEA2
QEB2
Pin header 6x1 2mm
C4 100n
C5 100n
C7 100n
Small number of decoupling capacitors as the modules have their own.
H1
VCC
GND
TXD
RXD
HC05

Drive-train development



AS5045 magnetic encoder

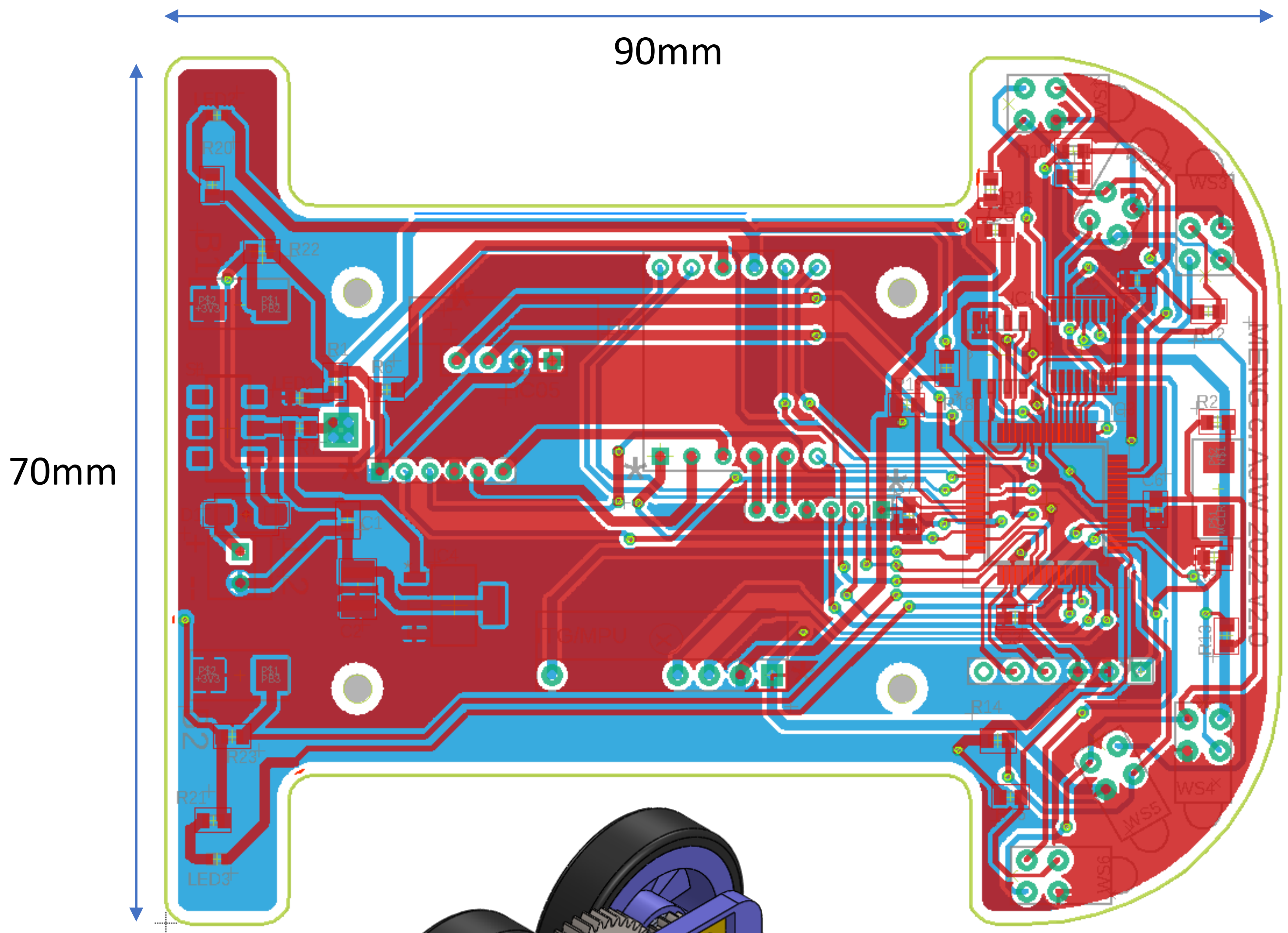


Magnetic encoder on extended shaft

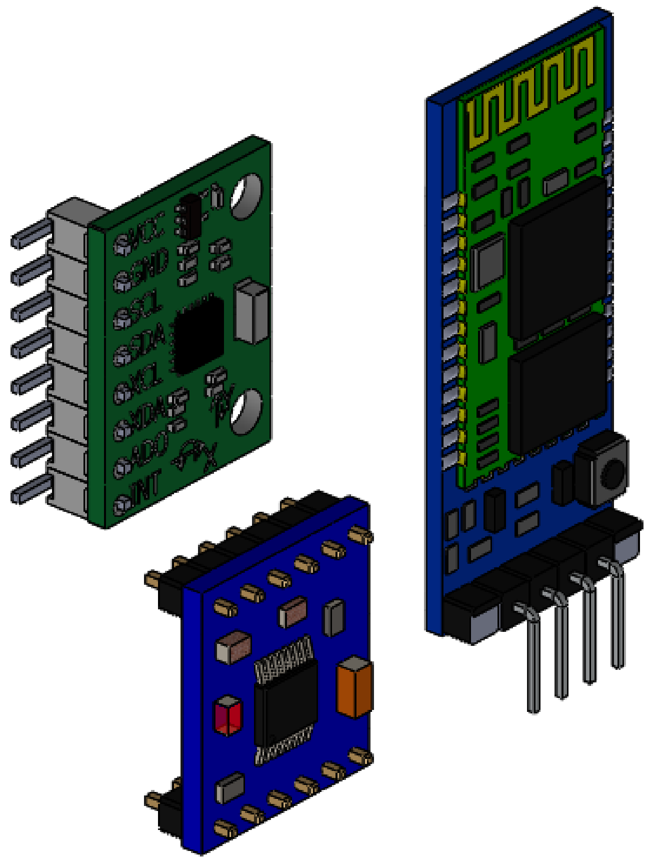
Double 8x4x3 mm bearings

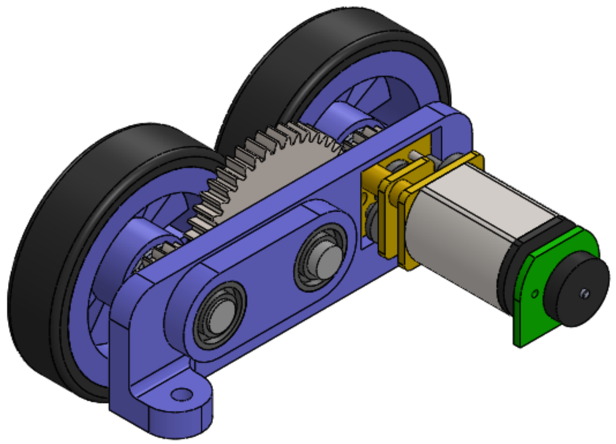
Larger Wheels, Larger central gear to give more clearance under the motor while maintaining ground clearance

MSRV2 (2014) $\longrightarrow$ MENG4A (2022)



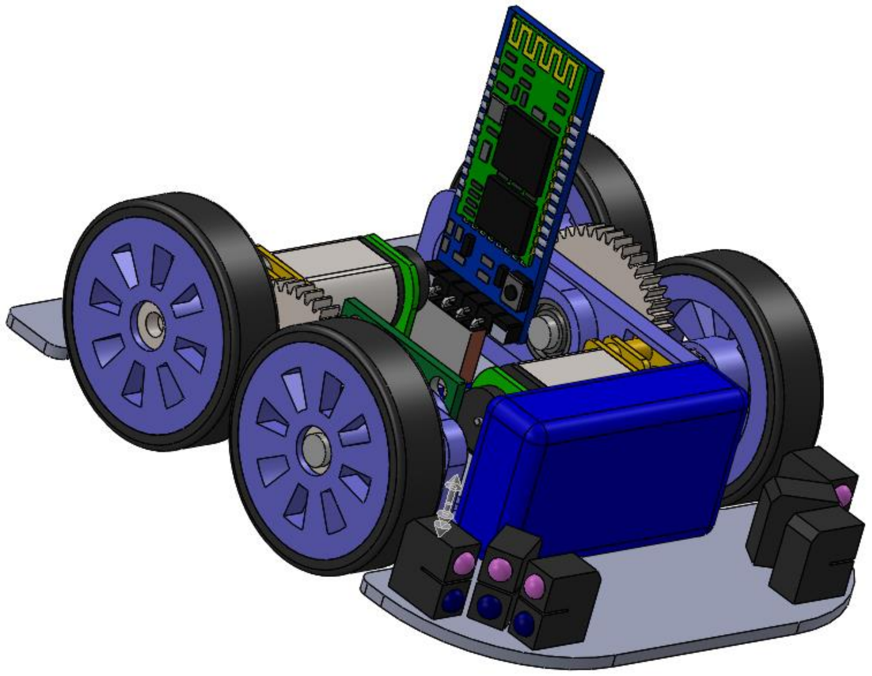
Gyro and Bluetooth modules optional





Drive train is stand-alone - can be replaced with a student design (Mech) or can be used on alternate base/pcb (Elec) dependent on the course it is used for – Mech/Elec.

Meng Electronic Engineering Group Project – 4th Year Student + staff support

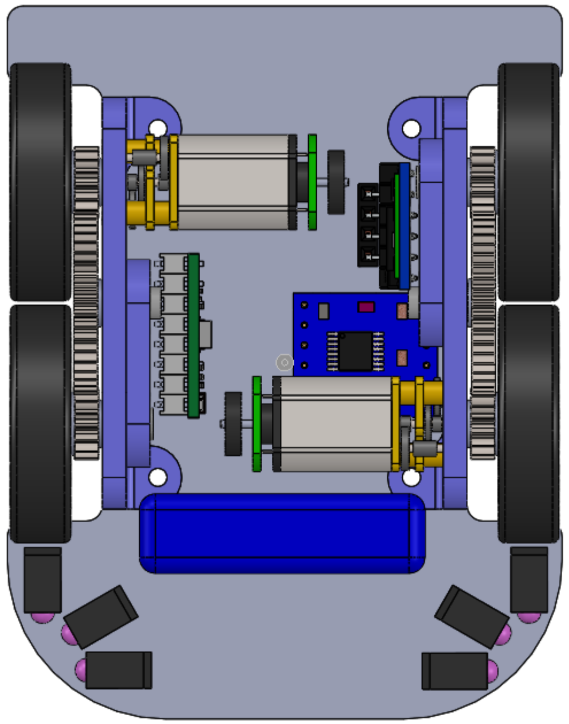


Designed as an MEng Project.

Use of off-the-shelf modules means more ‘up-front’ breadboarding options, simplified PCB layout, and smaller footprint of 70 x 90mm.

6 front sensors at 0,30,90 degrees
9V PP3 or 7V4/11V1 Lipo

FRAM/Bluetooth/RS232 data acq.
3-axis Gyro/Accelerometer

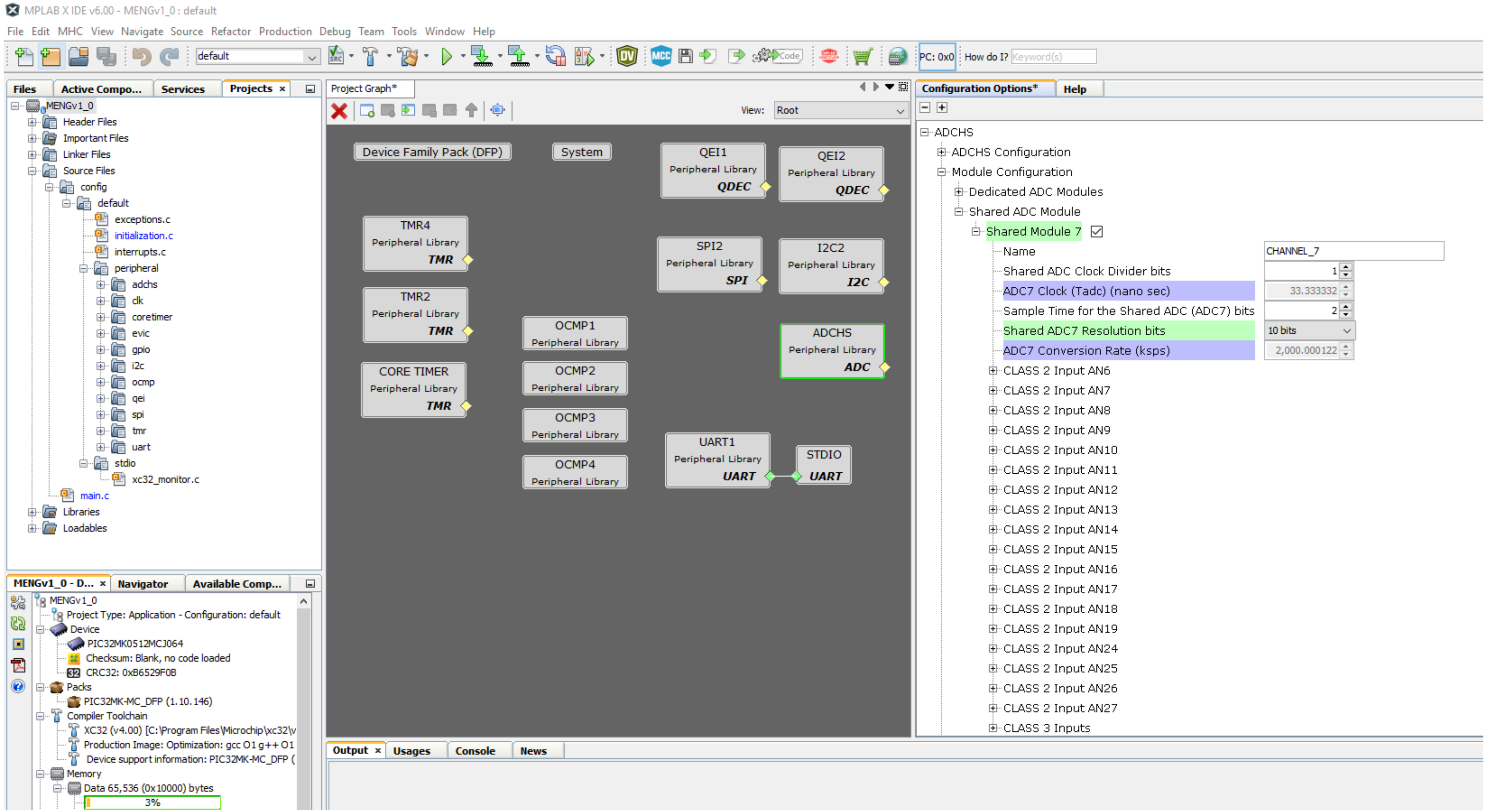


PIC32MK0512MCJ064 I/PT processor
MPLABX with XC32 compiler
Harmony3 code configurator.
Curiosity Pro Target board with PKoB4
PicKit3 or PicKit4 debugger

Interleaving N20 motors reduces width, allowing use of encoders on the extended shafts. Left and Right motor mounts identical. Several battery mount positions

MENG – 2022 Anthony Wilcox and Geraint Leahy

MPLABX with Harmony3



MPLAB X IDE v6.00 - MENGv1_0 : default

File Edit MHC View Navigate Source Refactor Production Debug Team Tools Window Help

default · PC: 0x0 · How do I? Keyword(s)

Files | Active Compo... | Services | Projects

- MENGv1_0
  - Header Files
  - Important Files
  - Linker Files
  - Source Files
    - config
      - default
        - exceptions.c
        - initialization.c
        - interrupts.c
        - peripheral
          - adchs
          - clk
          - coretimer
          - evic
          - gpio
          - i2c
          - ocmp
          - qei
          - spi
          - tmr
          - uart
        - stdio
          - xc32_monitor.c
    - main.c
  - Libraries
  - Loadables

MENGv1_0 - D... | Navigator | Available Comp...

- MENGv1_0
  - Project Type: Application - Configuration: default
  - Device
    - PIC32MK0512MCJ064
    - Checksum: Blank, no code loaded
    - CRC32: 0xB6529F0B
  - Packs
    - PIC32MK-MC_DFP (1.10.146)
  - Compiler Toolchain
    - XC32 (v4.00) [C:\Program Files\Microchip\xc32\v
    - Production Image: Optimization: gcc O1 g++ O1
    - Device support information: PIC32MK-MC_DFP (
  - Memory
    - Data 65,536 (0x10000) bytes
      - 3%

Project Graph* · View: Root

- Device Family Pack (DFP)
- System
- QEI1 Peripheral Library QDEC
- QEI2 Peripheral Library QDEC
- TMR4 Peripheral Library TMR
- TMR2 Peripheral Library TMR
- CORE TIMER Peripheral Library TMR
- SPI2 Peripheral Library SPI
- I2C2 Peripheral Library I2C
- OCMP1 Peripheral Library
- OCMP2 Peripheral Library
- OCMP3 Peripheral Library
- OCMP4 Peripheral Library
- ADCHS Peripheral Library ADC
- UART1 Peripheral Library UART
- STDIO UART

Configuration Options* | Help

- ADCHS
  - ADCHS Configuration
  - Module Configuration
    - Dedicated ADC Modules
    - Shared ADC Module
      - Shared Module 7 ☑

Name	CHANNEL_7
Shared ADC Clock Divider bits	1
ADC7 Clock (Tadc) (nano sec)	33.333332
Sample Time for the Shared ADC (ADC7) bits	2
Shared ADC7 Resolution bits	10 bits
ADC7 Conversion Rate (ksps)	2,000.000122

- CLASS 2 Input AN6
- CLASS 2 Input AN7
- CLASS 2 Input AN8
- CLASS 2 Input AN9
- CLASS 2 Input AN10
- CLASS 2 Input AN11
- CLASS 2 Input AN12
- CLASS 2 Input AN13
- CLASS 2 Input AN14
- CLASS 2 Input AN15
- CLASS 2 Input AN16
- CLASS 2 Input AN17
- CLASS 2 Input AN18
- CLASS 2 Input AN19
- CLASS 2 Input AN24
- CLASS 2 Input AN25
- CLASS 2 Input AN26
- CLASS 2 Input AN27
- CLASS 3 Inputs

Output | Usages | Console | News

Can this design be used for teaching Embedded Systems ?

- A 32-bit microcontroller with a lot of complex peripherals
- Documentation could be better (!)
- Harmony3 configurator to assist with processor and peripheral configuration
- A Curiosity Pro target board with issues
- Needs an external debugger PicKit3 (slow) or preferably PicKit4 (5x faster, 3x the price)
- A clunky IDE with a poor simulator.

What’s new here? We get used to the kinks! Of course this can be used for teaching Embedded Systems.

However - the success of a micromouse for teaching has little to do with the hardware.

Introspection

Integration of micromouse in teaching at any University in the UK will be due to the personal interest of academic staff – quite often an individual. That individual will need to have some control over programme content, as continuity from year to year at the same level, and ensuring appropriate underpinning at lower levels, is essential. Successful implementation also requires a high level of technical support.

Unfortunately, in my view and from experience, this is where it will only usually succeed in the short-term. Changes in staff, student intake and course modifications will all combine to phase it out.

I have had 20 years of Micromouse in University teaching – I think that might be a record ?

To close – some words from a guy who was teaching Advanced Embedded Systems, but who took on an introductory course also in Embedded Systems.

How I Teach Embedded Systems - Jay Carlson

“When I started thinking about what I would change about the class, one of the first things to go was the Arduino platform it was taught with.

I honestly have no idea why Arduino is used as much as it is in EE / CS embedded education programs, since it has such poor alignment with most learning outcomes for these classes:

•**Difficulty accessing dev environment internals:** how do I view the assembly code output from Arduino? Or the hex file that will get programmed? Can I control how variables are placed in RAM, or configure the linker or compiler optimization settings in any manner?

•**Difficulty accessing hardware internals:** there’s no debugger. It’s insane to teach an embedded systems course with a platform that does not allow you to set breakpoints or inspect or modify memory.

•**Strange, non-standard C preprocessor secret sauce** *(sic):* “void main()”? Nah. Call functions without declaring them first? Sure. Wait, what headers are included by default? Who knows. Students leave the course thinking that *DDRB*, *Serial.print()* and *uint16_t* are all reserved words in C that you can use anywhere.”

How I Teach Embedded Systems - Jay Carlson

“My other (more pragmatic) issue with the Arduino platform is the nonstandard tooling. There are tons of MCUs students will encounter professionally. These generally all work the same:

You write software in C, and then you compile it, getting some sort of hex file.
You use a debugger or a programmer to communicate with the MCU and load the program code into the flash memory of the microcontroller, and then you run it.
Most of the time, you have a debugger attached that can set break points, inspect memory, and receive trace data.

The Arduino Uno ecosystem is not representative or similar to any other microcontroller ecosystem, so it seems like a bizarre choice.”

I feel the same way …….

Also well worth a look is: *Embedded FM Podcast*

END

“I cannot teach anyone anything, I can only make them think.” - Socrates