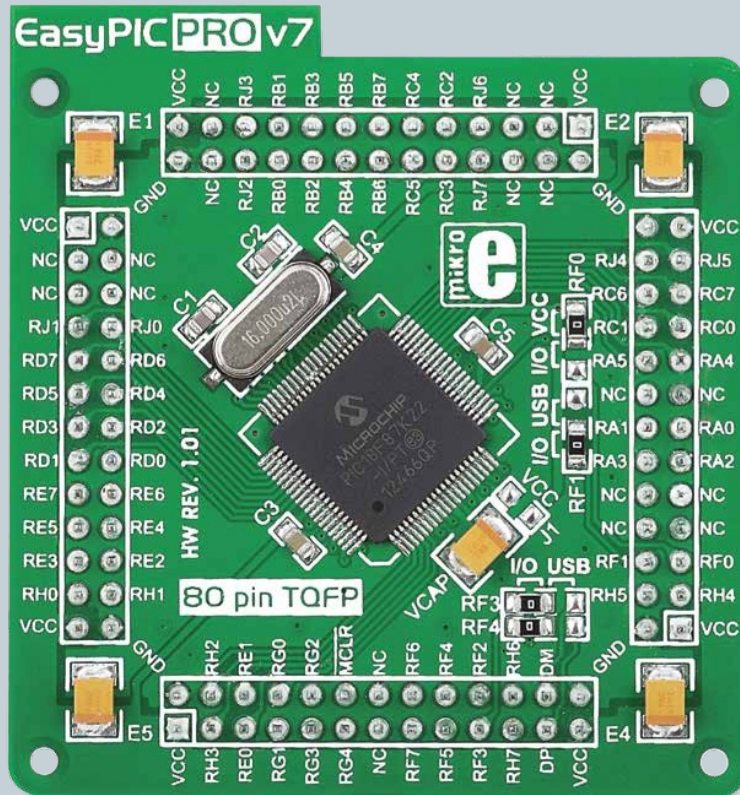


# Microprocessor Experiment G1

1

**TUESDAYS 9:00-12:00**  
**THURSDAYS 9:00-12:00**  
**FRIDAYS 9:00-12:00**

# The PIC18F87K22 microcontroller



- 8-bit microprocessor
  - Does calculations 8-bits at a time
- 64 MHz clock
  - 1 instruction every 4 clock cycles
- Limited memory
  - 128K flash program memory
  - 4K RAM data memory
- Lots of useful peripherals
  - Comms, ADC, Timers, digital I/O
- 10s of billions of this sort of microprocessor shipped per year
  - They are everywhere!
  - And they are extremely useful in the physics lab

# You've seen this - Python

3

```
def _find_penrose(pos):
    n = len(pos)
    sep = np.zeros((n, n))
    for i in range(n):
        sep[i,:] = np.hypot(*(pos - pos[i]).T)
    dmid = np.partition(sep.ravel(), int(3.5 * n))[int(3.5 * n)]
    dlow = 0.91 * dmid
    dhi = 1.09 * dmid
    neighbourcount = np.sum((sep > dlow) & (sep < dhi), 0)
    indices = np.flatnonzero(neighbourcount == 6)
    pos6 = pos[indices]
    c_offset = np.empty_like(pos6)
    for i, idx in enumerate(indices):
        neighbours = np.flatnonzero((sep[idx,:] > dlow) & (sep[idx,:] < dhi))
        c_offset[i] = np.sum(pos[neighbours] - pos[idx], axis=0)
    return (dmid, pos6, c_offset)
```

# You've possibly seen this – C++

4

```
float clarity_processor::find_penrose(std::vector<cv::Point2f> pts, std::vector<cv::Point2f> &pts6, std::vector<cv::Point2f> &centroid6){
    size_t n = pts.size();
    std::vector<float> distance(n*n);
    for (size_t i=0; i<n ; i++) {
        for (size_t j=0; j<n; j++) {
            distance[i*n+j] = hypot(pts[i].x-pts[j].x,pts[i].y-pts[j].y);
        }
    }
    std::vector<float> distance_sorted = distance;
    std::partial_sort(distance_sorted.begin(),distance_sorted.begin()+4*n,distance_sorted.end());
    float dmid=distance_sorted[(int) (3.5*n)];
    std::vector<int> neighbours(n);
    std::vector<cv::Point2f> neighbour_centroid(n);
    for (size_t i=0; i<n ; i++) {
        int neighbours=0;
        cv::Point2f centroid(0.0,0.0);
        for (size_t j=0; j<n; j++) {
            if (distance[i*n+j] > 0.91f*dmid && distance[i*n+j] < 1.09f*dmid) {
                neighbours++;
                centroid.x += pts[j].x-pts[i].x;
                centroid.y += pts[j].y-pts[i].y;
            }
        }
        if (neighbours == 6) {
            pts6.push_back(pts[i]);
            centroid6.push_back(centroid);
        }
    }
    return dmid;
}
```

# Programming - Assembler

5

```
SET_DUCYC
    ;subroutine to set duty cycle value
    andlw    0x03
    ;sets max ducyc value to 3 (0.3usec)
    movwf    DUCYC
    btfss    DUCYC,0
    goto     CLEAR2Y
    bsf      CCP2CON,CCP2Y
    goto     CONT2Y
CLEAR2Y
    bcf      CCP2CON,CCP2Y
CONT2Y
    btfss    DUCYC,1
    goto     CLEAR2X
    bsf      CCP2CON,CCP2X
    goto     CONT2X
CLEAR2X
    bcf      CCP2CON,CCP2X
CONT2X
    clrc
    rrf      DUCYC,W
    movwf    TEMP
    clrc
    rrf      TEMP,W
    movwf    CCPR2L
    return   ;end of subrouting Setducyc
```

- You will probably never have seen anything like this before
- It takes skill to program anything complicated
- But it lets you into the intimate inner workings of a microprocessor
  - Control what it does at every clock cycle
  - Control all its physical functions

# The Course Goals are:

6

- To explain the inner workings of a computer at a fundamental programming level (assembler) with hardware interfacing.
- To teach you how to work independently and find for yourself everything you need for your project.
  - Hence, no book. Everything you need is on the Web
  - Blackboard and Teams
- This is a lab course, not a lecture course
  - As the lab progresses, you will be expected to work more independently
- To train you to design, construct and document a commercial product
- To give you insight into the use of microprocessors which have a wide range of applications in automobiles, appliances and other industrial applications as well as tools in research laboratories

## Microcontrollers in the real world

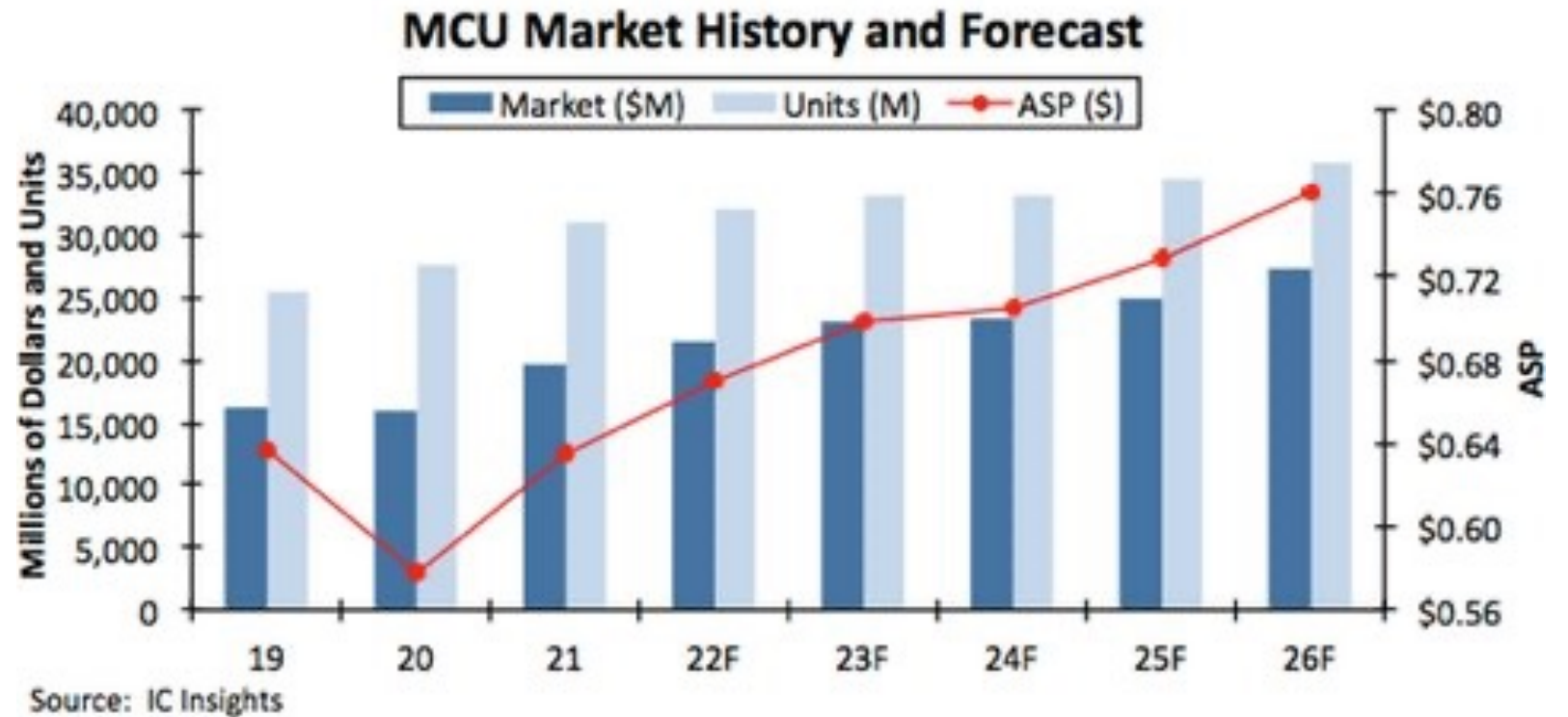
A microcontroller is basically what we call a small microprocessor that is designed with relatively limited functionality compared to a desktop computer

It might have lots of useful simple hardware peripherals built into it though

Basically you will find them everywhere.

Projected sales for 2026 are more than 4 microcontrollers for each member of the Earth's population

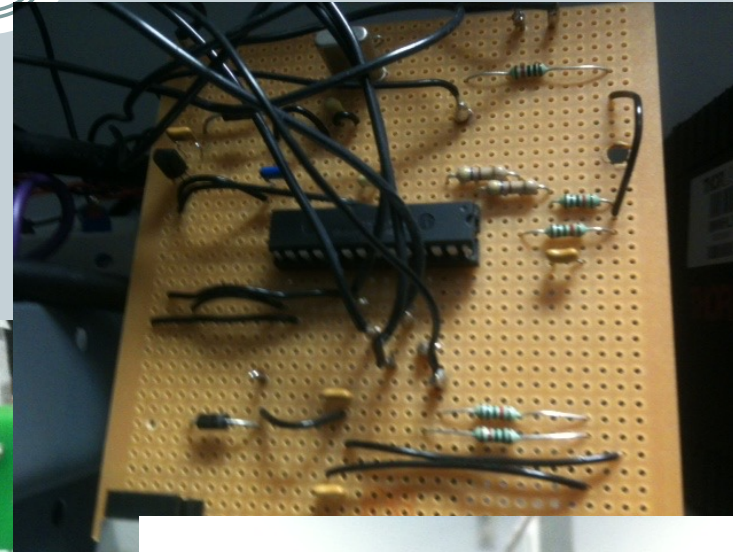
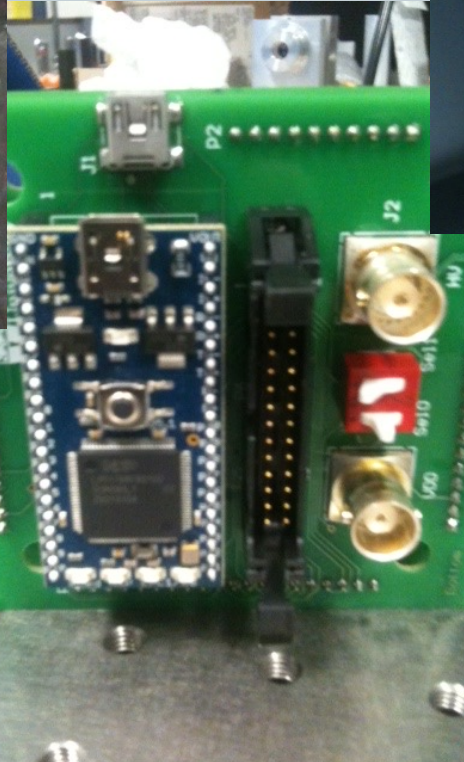
Note the price!





# Microcontrollers in the research lab

8





# Skills you will learn in the lab

9

- Designing basic electronic circuits and interfaces
- Understand how a microprocessor functions and how to use one.
  - We will use the Microchip PIC18F87K22 microprocessor
- Programming in the microprocessor's most basic language (PIC18 assembler) including
  - the basic assembly instructions
  - writing a structured program
  - using tools to compile and download your programs to the PIC18 chip
  - use Git and Github for source code control and collaboration
- You will probably never use Assembler again – but you will appreciate a lot about how higher-level languages and systems work by having done so

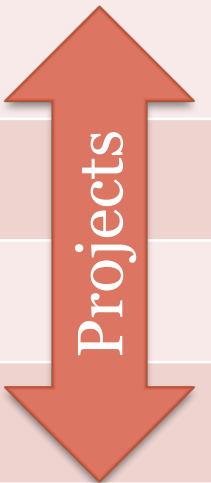
# Course Duration and Milestones

10

- About 3-4 weeks of lectures/training
- You will need to produce a short 2-page project proposal that will be marked for cycle 2.
- 3-4 weeks for project completion.
- Review meeting at the end of cycle 3
- Your written report for your project is due at the beginning of next term.

# Timetable

Term Week	Lectures released
Week 5: 30 <sup>th</sup> October	<ol style="list-style-type: none"> <li>1. Introduction</li> <li>2. Assembler 1</li> <li>3. Assembler 2</li> </ol>
Week 6: 6 <sup>th</sup> November	<ol style="list-style-type: none"> <li>4. External electronics</li> <li>5. Serial to parallel</li> <li>6. Liquid crystal display</li> </ol>
Week 7: 13 <sup>th</sup> November	<ol style="list-style-type: none"> <li>7. Keypad</li> <li>8. Analog to digital converter</li> <li>9. Timers and Interrupts</li> </ol>
Week 8: 20 <sup>th</sup> November	<ol style="list-style-type: none"> <li>10. Introduction to projects</li> </ol> <p>Proposal hand-in 23<sup>rd</sup> November</p>
Week 9: 27 <sup>th</sup> November	
Week 10: 4 <sup>th</sup> December	
Week 11: 11 <sup>th</sup> December	<p>Project closeout</p> <p>Project review on Friday</p>
Term 3:	Report hand-in 9 <sup>th</sup> Jan



# Time-keeping

13

We will need to move quickly through the taught exercises to allow enough time for completion of your project work

- Don't get hung up on completing every part of every exercise
- Often there are suggestions of things that you can try out, but only do this if you have time
- Concentrate on completing the core parts of each exercise if you can

# The course is assessed on the projects you create

14

- You will use a microprocessor to create an application and construct something useful :
  - Some electronics will be needed
    - ✦ You will work through building some electronics in the first weeks of the lab
  - You will learn to interface the microprocessor with various devices
    - ✦ There is a wide selection of possible devices to connect to the microprocessor
  - You will make a new product of your choice
    - ✦ Many interesting inventions have been produced in the lab in previous years
- You will need to write a good and clear project proposal for cycle 2 and then a report on the product you have made for cycle 3, and demonstrate the functioning of your product
- Your mark in this lab will be based on both your proposal and your final report
  - The exercises we do in the first weeks are to enable you to undertake a project on your own
  - You should record these exercises in your lab book as we go along



# Lab books

15

- You will need to work efficiently to complete on time
  - You will need to keep a record of what you are doing for...
    - Discussions with your demonstrator to help you explain what you are doing
    - So you can write a good report at the end
- THIS IS WHAT LAB BOOKS ARE FOR!**
- Talk to your demonstrator regularly about your progress and show them your lab book
    - They will give you advice on what you need to record

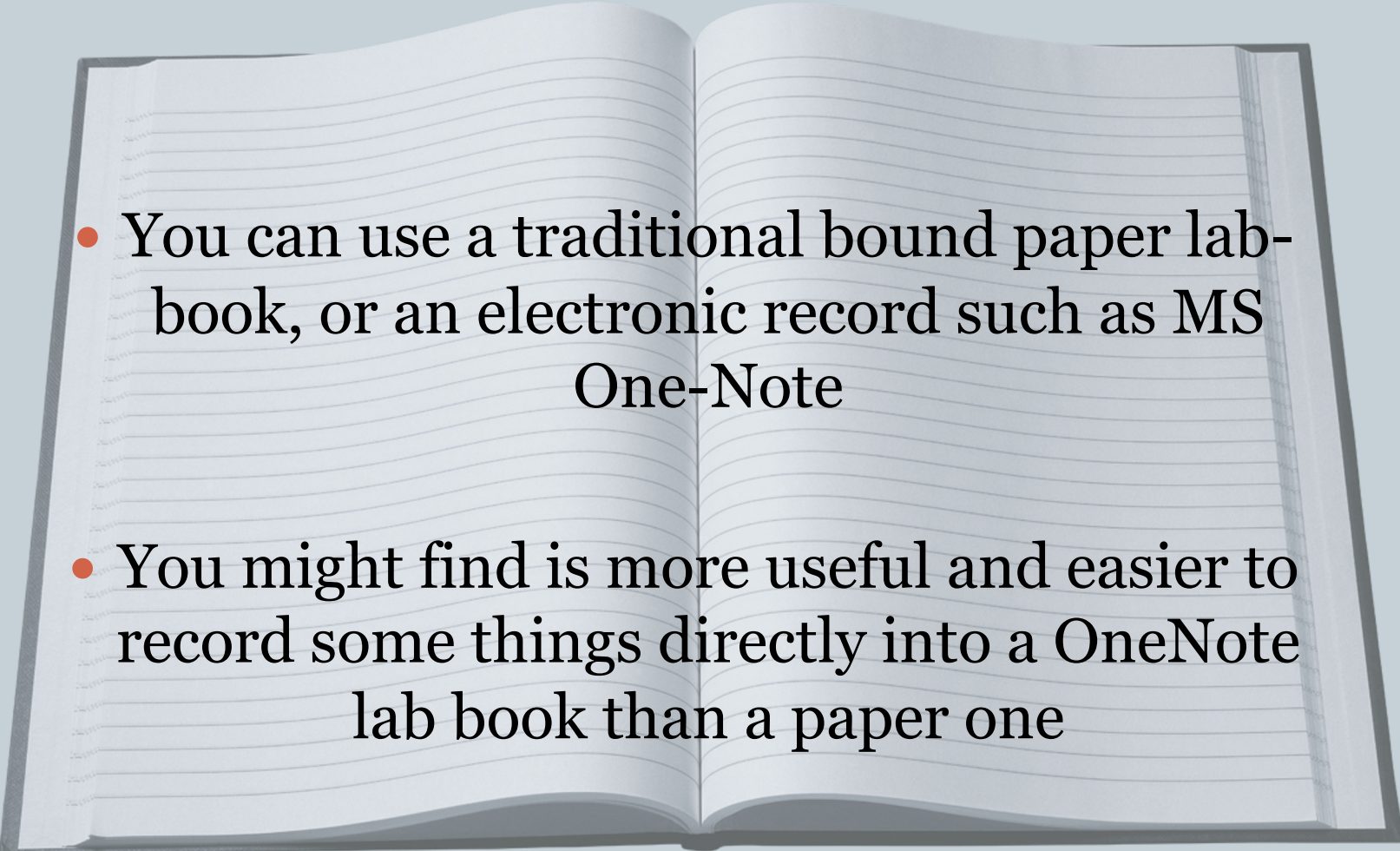
# What to put in your lab books

16

- Start a new page for each session – and date it
- Note what you are aiming to achieve in that session
  - Add updates to your aims as you progress
- Sketch out your ideas directly into your lab book:
  - Circuit diagrams (schematics)
  - Top-down modular diagrams for the structural bigger picture
  - Flow charts for the internals of your code
  - Snippets of code
- Use git/github to “control” your source code development
  - Note in your lab book where/why/when you make branches, commits or merges
  - Record the commit message and the revision number (eg fdb2298)
- Record any data measurements you might make
  - Numerical measurements and any graphs from those measurements
  - Print-outs of oscilloscope traces (you can get these onto a usb stick or into your computer by usb)
- Always stick loose sheets into your lab book
- At the end of your session, briefly summarise your progress, its successes (and failures)
  - Perhaps add suggestions of what to do next

# Lab books

17

- 
- You can use a traditional bound paper lab-book, or an electronic record such as MS One-Note
  - You might find it more useful and easier to record some things directly into a OneNote lab book than a paper one

## Demonstrators

Demonstrators will be available according to the table to the right

Demonstrators are here to provide you with guidance

But in third year lab, do not expect them to provide you with all the answers

Use your lab books to help in your discussions with demonstrators

	Demonstrators
Tue	Kirika Uchida
Thu	Bartosz Krawczyk
Fri	Raghu Shukla