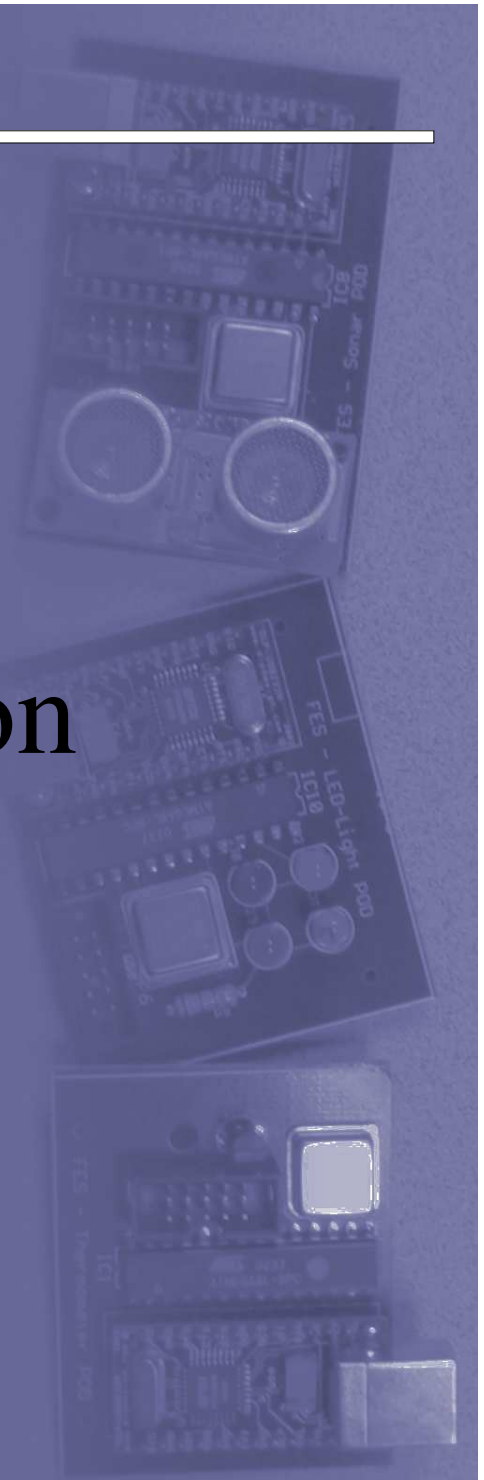
PODs: Physical Object Devices

Thesis Defense presentation for Frank Sorenson

February 19, 2004



Motivation

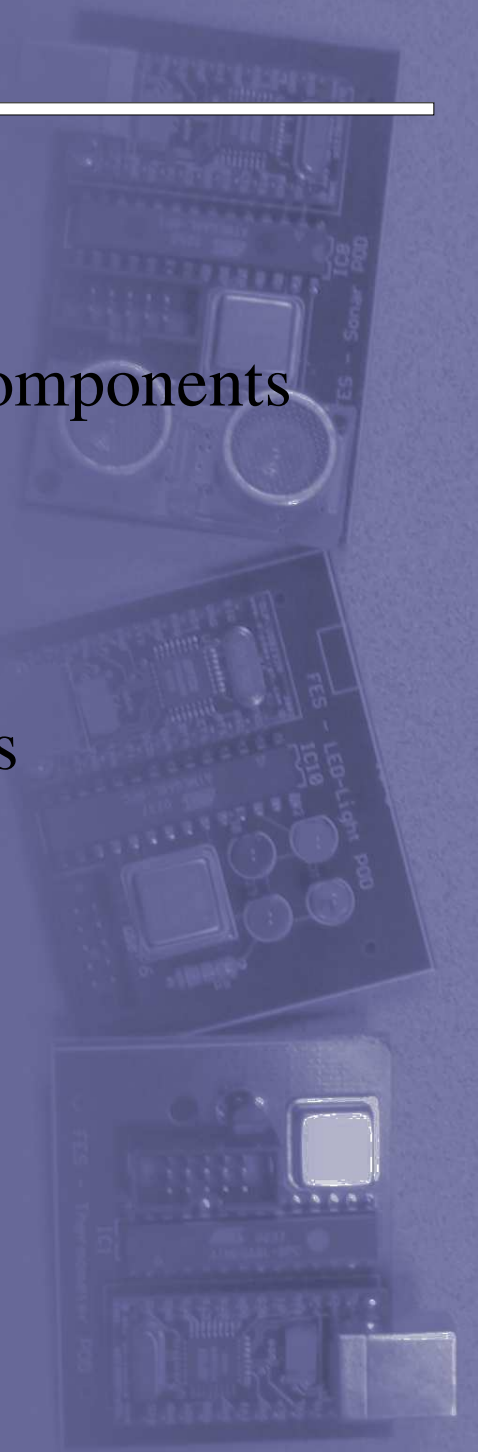
- Programmers are often interested in controlling and using electronic devices in their work
- Computer control of electronic devices can be difficult
- Many programmers are more interested in controlling and using devices, not constructing them
- Skills lie in programming, not hardware design

Difficulties controlling electronics

- Many different hardware interfaces makes choice and compatibility difficult
- Some devices utilize specialized, proprietary software
- Some devices do not have computer-usable interfaces
- Time is often spent (wasted) designing and maintaining electronics
- Solutions are often not flexible or extensible

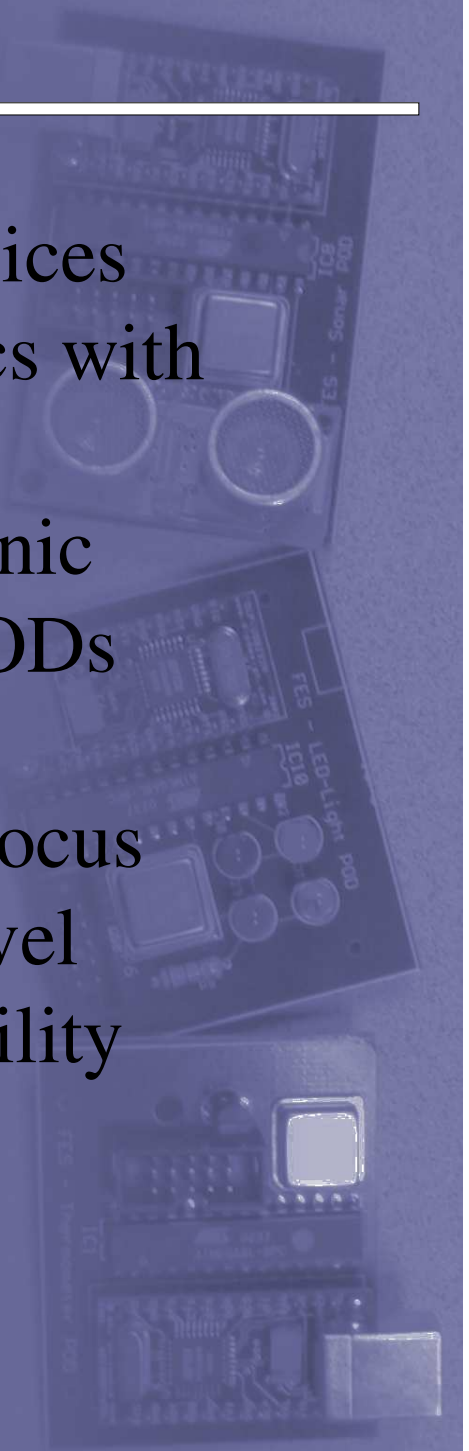
Previous work

- OOpic
  - Programmable device can act as various components
  - Low-level interface
- Lego Mindstorms
  - Programmable, popular, many sensor types
  - Limited input and output, hobby
- Phidgets
  - Help abstract physical devices
  - Not lightweight interface



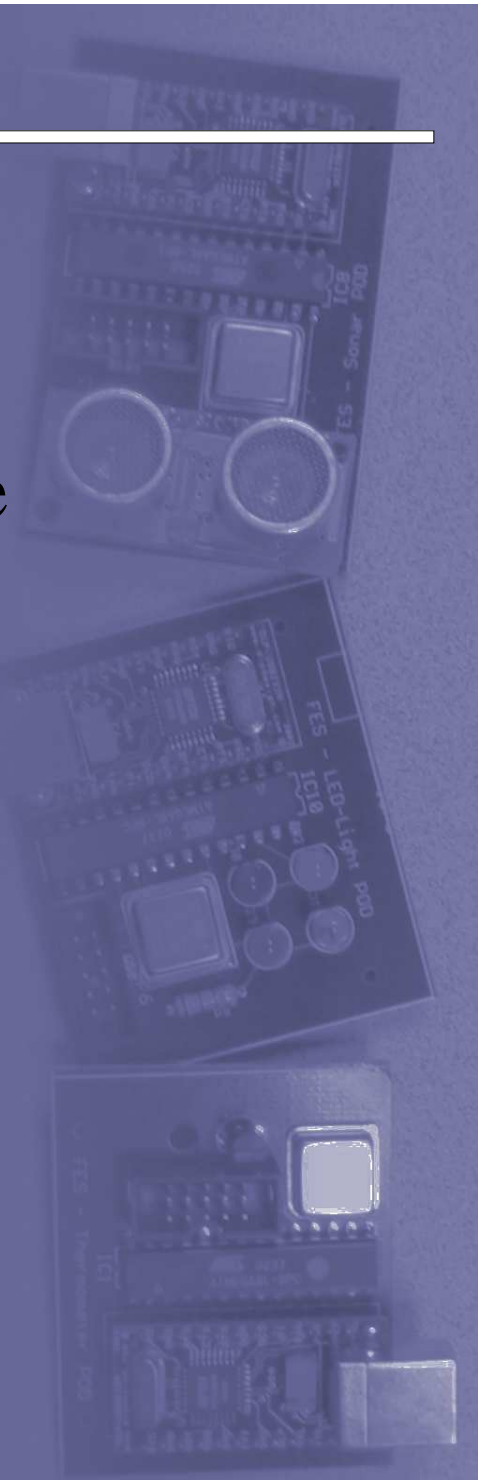
Our Thesis Statement on PODs

- By building intelligence into electronic devices themselves, and by designing the electronics with a common hardware interface and software library, programmers can easily use electronic devices directly in programs they write. PODs provide an object-oriented programming framework that allows the programmer to focus on the use of the device, rather than low-level details. This work demonstrates the feasability and usefulness of this design method.



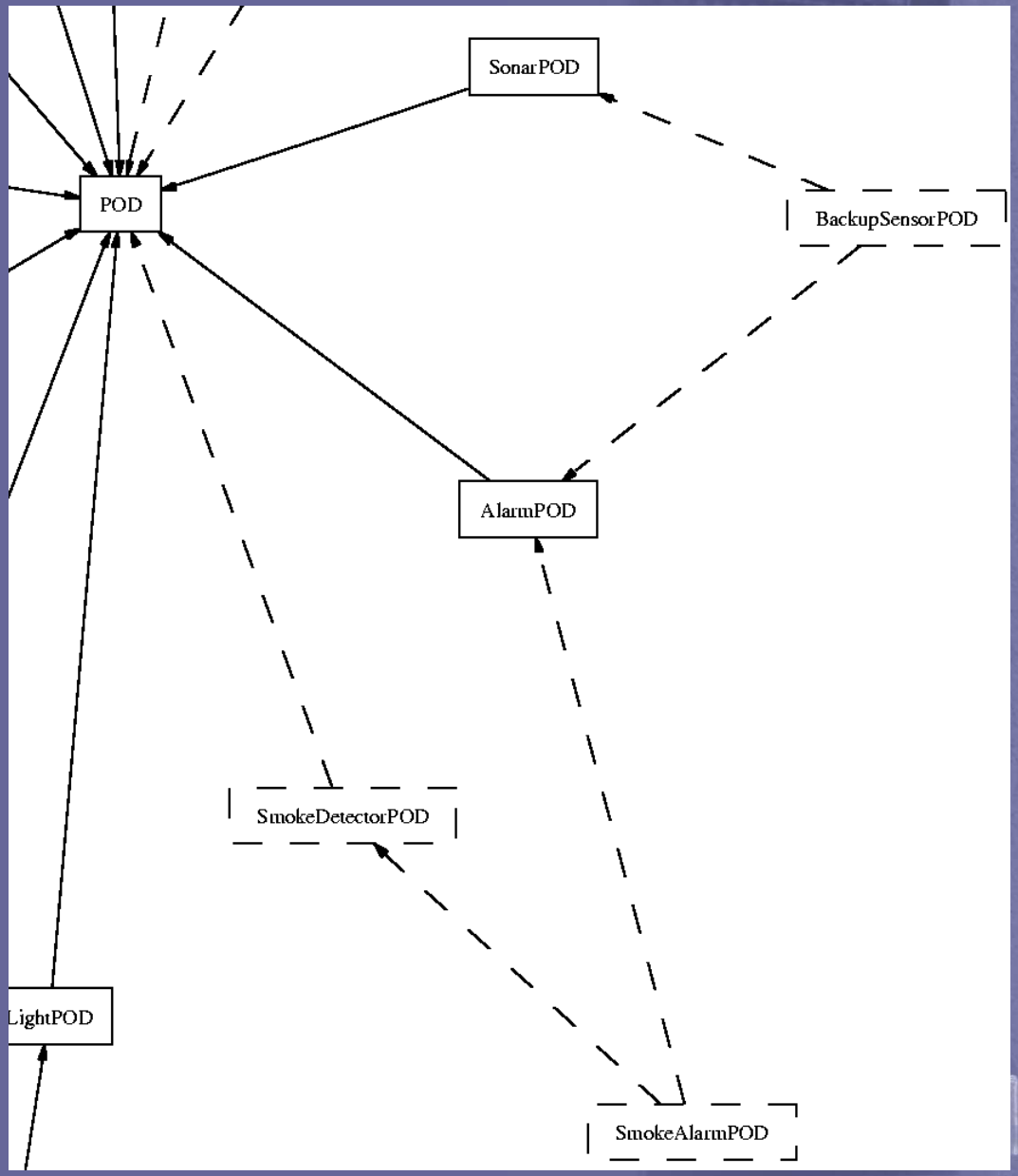
PODs: The Vision

- Hierarchy of object-oriented hardware devices
- Programming environment treats hardware as software
- Software library to simplify programming
- Hardware interface is standardized
- PODs can retain settings or be assigned unique identifiers
- Little or no electronic knowledge is required



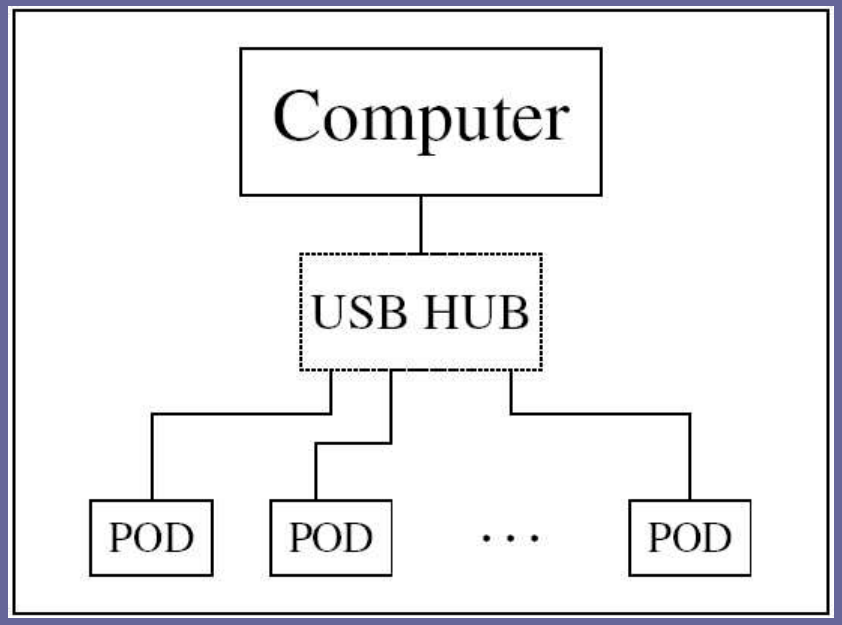
Hierarchy of hardware/software objects

- POD is a base class and has certain standard properties
- Each POD type derives from other POD classes and adds their own unique features
- Programmers instantiate hardware objects in their programs
- Complex hierarchies of POD classes are possible



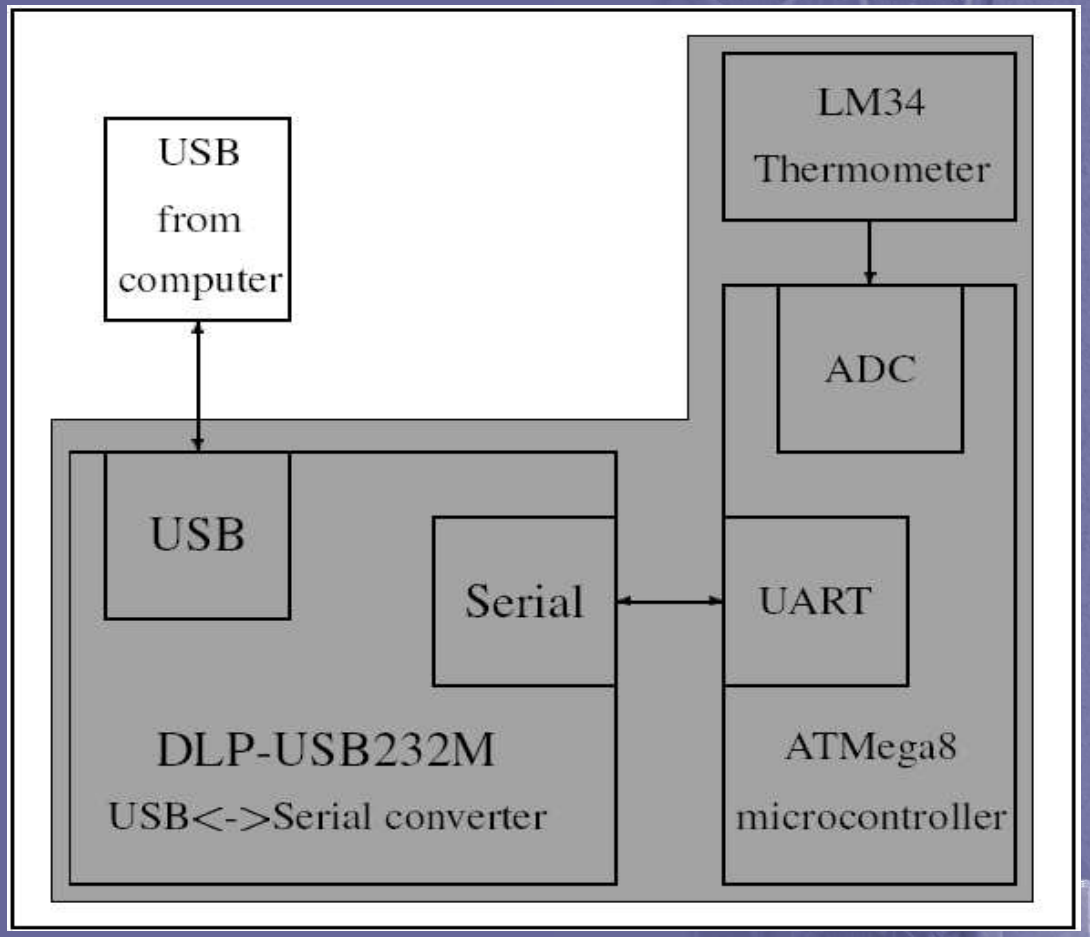
POD Hardware Interface - USB

- USB is a common interface
- Power and Data via the same wires
- Expandable to 127 devices



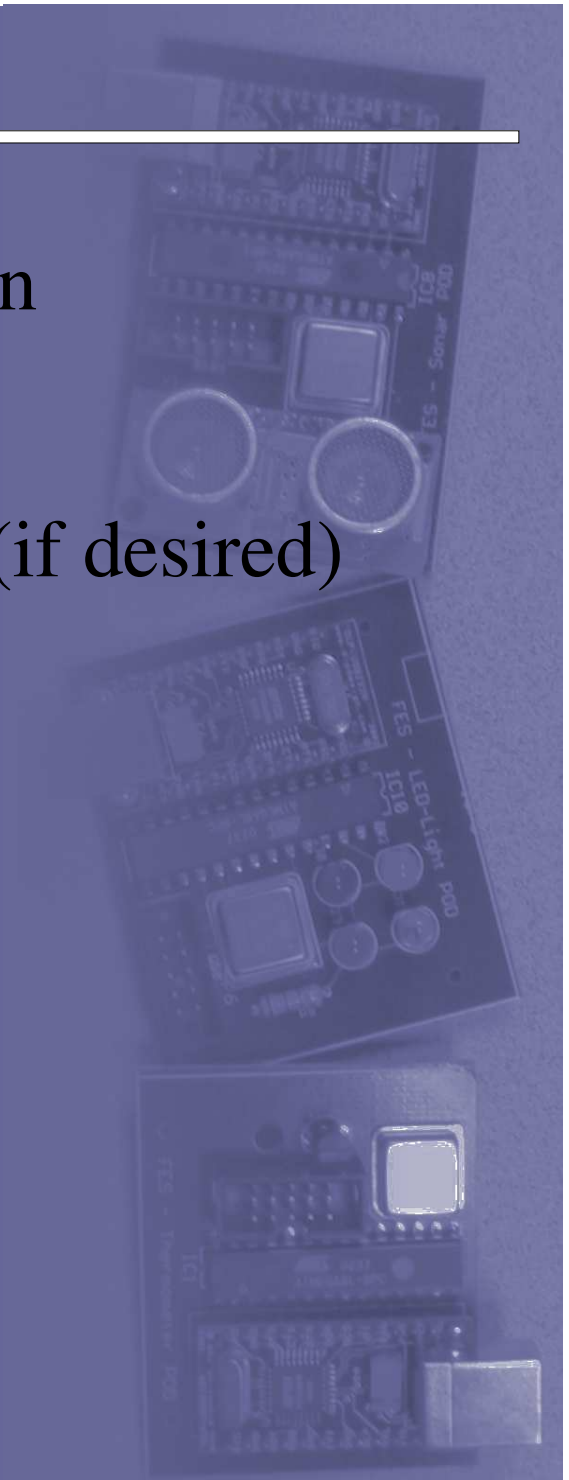
Block diagram of each POD

- USB interface
- USB<->Serial connector
- Onboard microcontroller
- Additional electronics

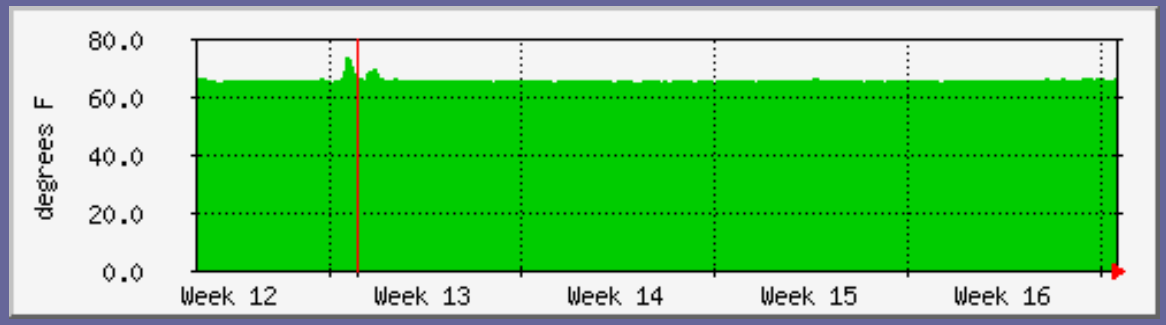


POD Programming and interaction

- Programmer chooses POD by function
- Attach POD to system
- Configure POD settings or identifier (if desired)
- Write simple program
- Compile and link with POD library
- Run application



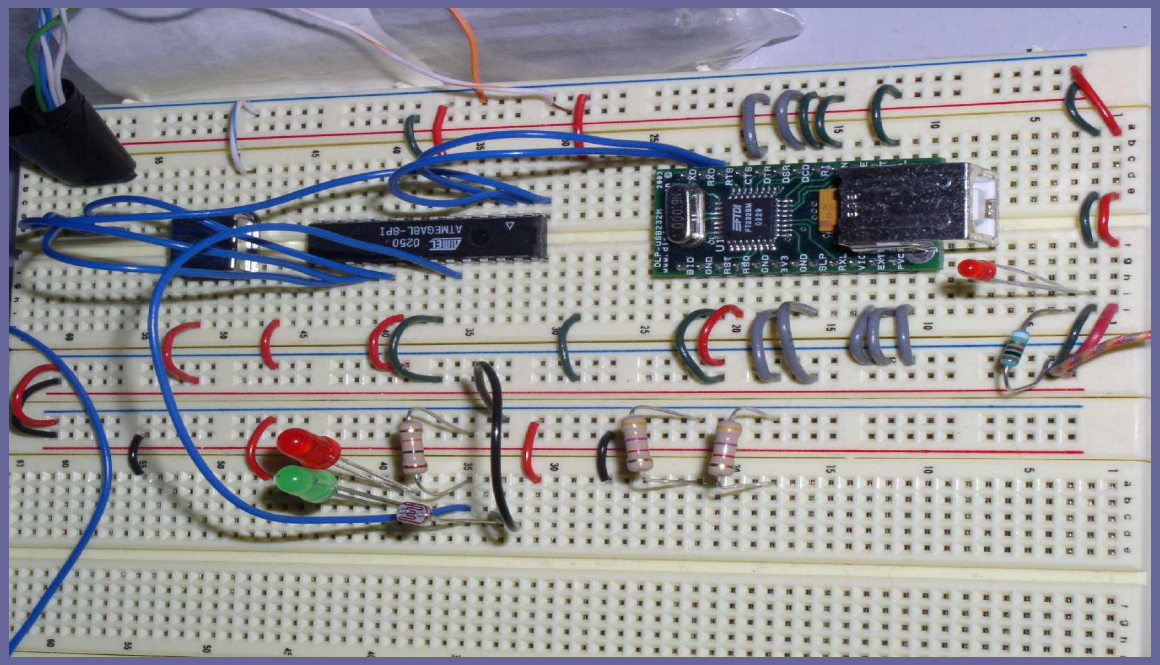
Simple programming – Thermometer POD




```
#include <stdio.h>
#include <libPOD.h>

int main(int argc, char *argv[])
{
   PODEnv *MyPODs;
   ThermometerPOD *Thermometer;
   float Temperature;

   MyPODs = new PODEnv();

   Thermometer = dynamic_cast<ThermometerPOD *>
      (MyPODs->FindPODbyType(POD_TYPE_THERMOMETER));
   if (Thermometer != NULL)
   {
      Thermometer->GetTemperature(&Temperature);
      printf("\%f", Temperature);
   }
   else printf("0"); /* if no Thermometer POD is connected */
   printf("\n0\n0\n0\n");
}
```

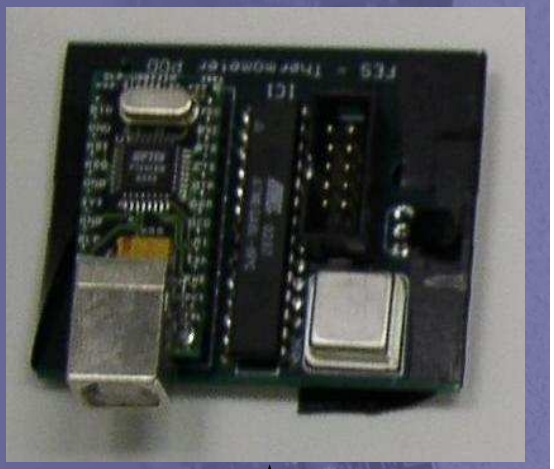
Building a POD – Initial design

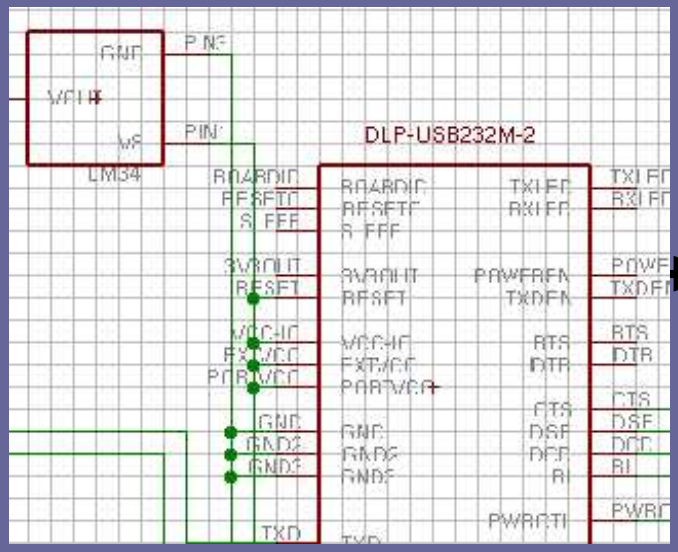
- Determine functions POD will perform
- Breadboard possible designs
- Choose components and design

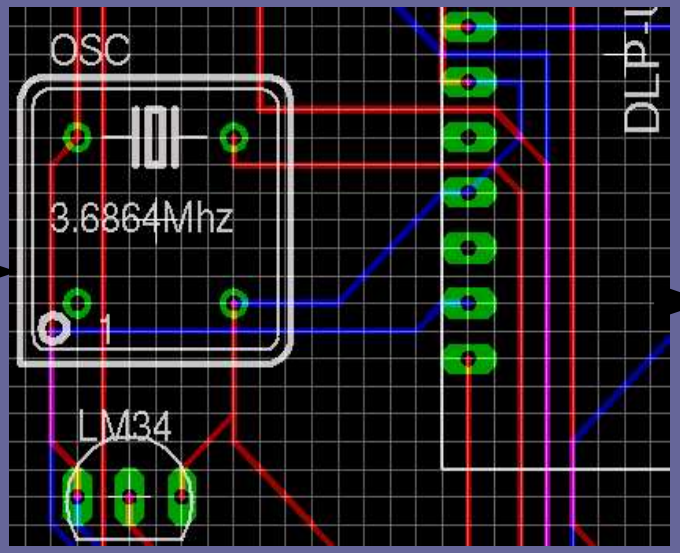


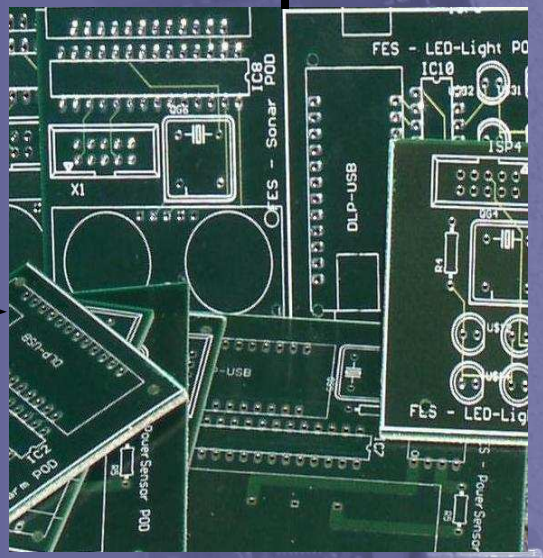
Building a POD – Hardware construction

- Create schematic
- Design PCB based on schematic
- Manufacture and build POD





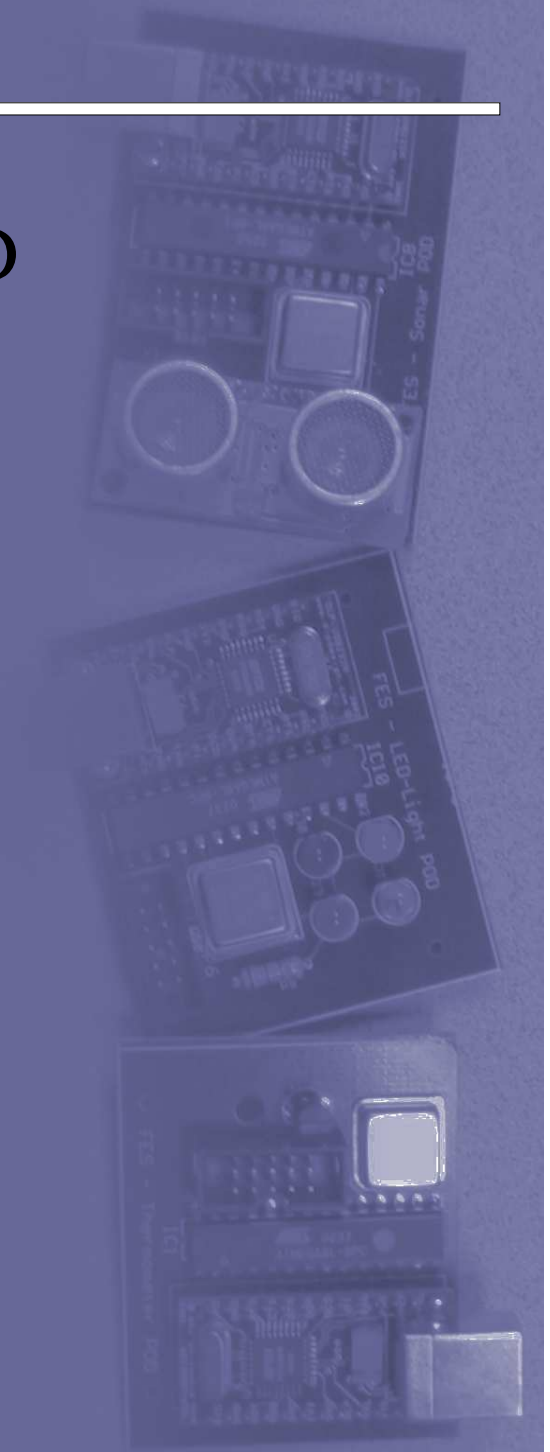




Building a POD – Software

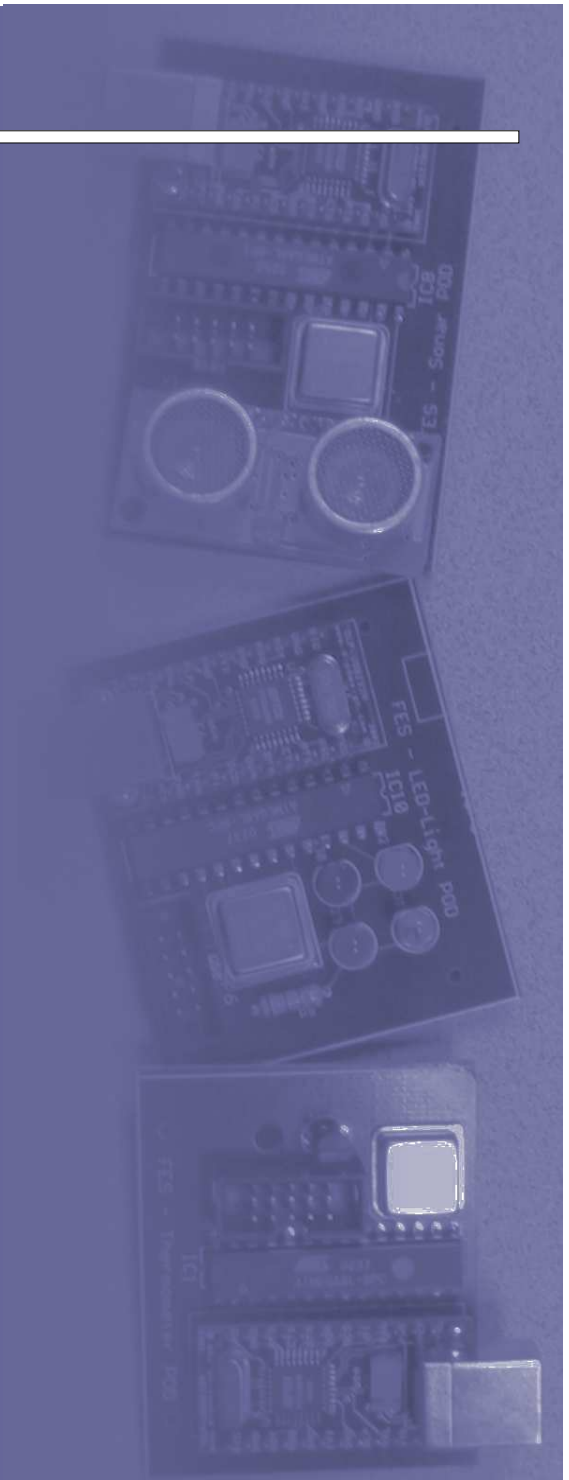
- Program microcontroller and test POD
- Write POD software interface class

```
#ifndef __THERMOMETERPOD_H__
#define __THERMOMETERPOD_H__

#include "PODDefines.h"
#include "POD.h"

class ThermometerPOD : public POD
{
   public:
      ThermometerPOD();
      ThermometerPOD(int DeviceNum,int PortNum);
      ThermometerPOD(int DeviceNum);
      int GetTemperature(float &ReturnTemp);
      int GetUnits();
      int SetUnits(char Units);
      virtual int Test();
   protected:
      int ProcessAlert();
   private:
};

#endif /* __THERMOMETERPOD_H__ */
```



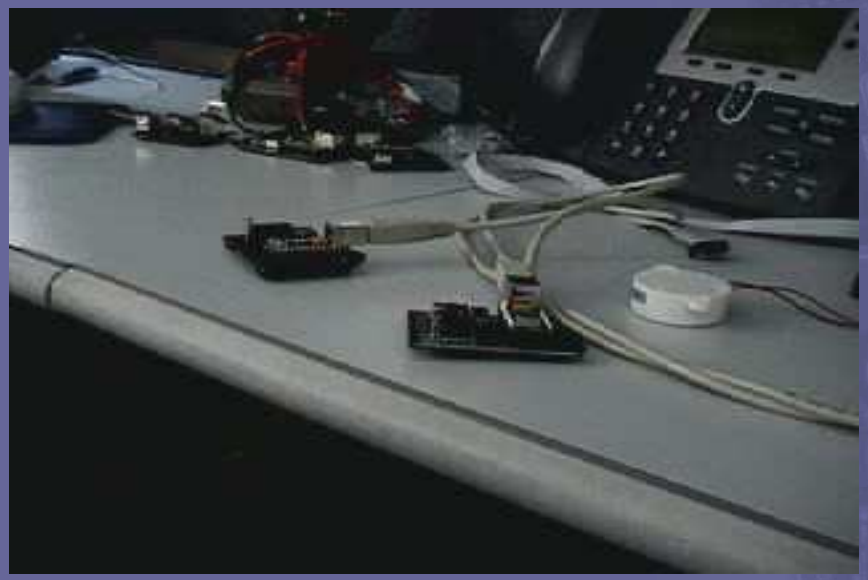
PODs implemented

- Thermometer POD
- Light Sensor POD
- Compass POD
- Motor Control POD
- Buttons POD
- Light Emitting Diode (LED) POD
- Sonar Distance Sensor POD
- Alarm POD
- Power Sensor POD



Demo – Light sensors

- Multiple light sensors require unique identification
  - PODEdit utility
  - Assign PODs 'Left' and 'Right'
- Alarm indicates which sensor is measuring greater light
  - 1 beep Left
  - 2 beeps Right



Demo – Robot movement

- MotorControl PODs cause robot to turn to each side and go forward and backward

```
while ((CurrentTime - StartTime) < 60.0)
{
   MyRobot->Right();
   usleep(1000000);
   MyRobot->Forward(50);
   usleep(500000);
   MyRobot->Backward(50);
   usleep(500000);
   MyRobot->Left();
   usleep(1000000);
}
MyRobot->Stop();
```

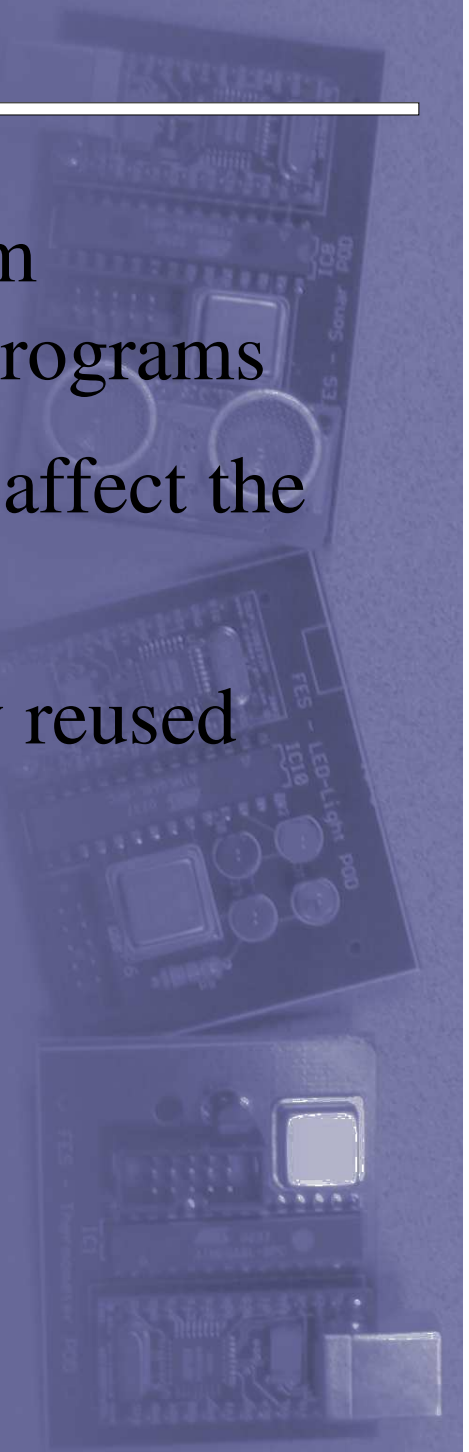
Demo – Robot finding North

```
class Robot : public PODEnv
{
   public:
      Robot();
      int Forward(unsigned char Speed);
      int Backward(unsigned char Speed);
      int Left();
      int Right();
      int Stop();
      float Heading();

   protected:
      MotorControlPOD *LeftMotor;
      MotorControlPOD *RightMotor;
      CompassPOD *Compass;
. . .
```

```
  MyRobot = new Robot();

   CurrentHeading = MyRobot->Heading();
   MyRobot->Right();
   while ((CurrentHeading > 10.0) && (CurrentHeading < 350.0))
   {
      CurrentHeading = MyRobot->Heading();
   }
   MyRobot->Stop();
```

Conclusions – POD benefits

- PODs are object-oriented, have a uniform interface, and are easy to integrate into programs
- Programmers make fewer decisions that affect the electronics
- PODs are flexible and hardware is easily reused in other projects



Conclusions – Possible Future Work

- Minimize and Miniaturize PODs
- Build additional interesting devices
- Investigate object-oriented properties further
- Improve hardware and software used with PODs
- Investigate additional hardware interfaces
- Web-enabled PODs

Final Remarks

- We designed an object-oriented programming environment that allows simple control of electronic hardware
- We demonstrated the use of PODs through case studies
- We built intelligence into the PODs themselves, developing a useful method for computer control of hardware

Questions?