# System Design Philosophy:
# An integrated approach

1. System Design Considerations
2. System Architecture Diagrams
3. Communication Protocols

# Systems Design - Introduction

A system is defined as a set of interconneted components that work together to achieve a complex outcome.

Systems design approaches are common in Mechatronic products, as there natural interfaces between mechanical, electrical, and software components.

# Electronic Systems Design - Reflection

Electronic Systems Design will be used as an example throughout this lecture series.

## Self-Reflection

What is your current practice for designing electronic systems (your design philosophy / process / algorithm)?

What manufacturing processes and technologies are you currently familiar / comfortable with (eg. Throughhole, Surface Mount, Breadboard, Vero, Etched/milled PCB)?

What are the advantages and limitations of your current techniques?

How do you address future-proofing in your designs?

# Systems Design - Introduction

1. System Design Considerations
   - Needs Analysis (Specification)
   - Design for contingency / future-proofing
   - Design for Modularity / Scalability
   - Design for Manufacture
2. System Architecture Diagrams
   - Node Layout and Information Flow
   - Relationship Diagrams
3. Communication Protocols
   - Different Transfer Methods (Including Serial/Parallel)
   - General Challenges
   - Addressing Methods
   - Advantages and Limitations

# Systems Design Considerations - Mobile Phone Example

What does a mobile phone consist of? What are the requirements / features / expectations?

How can a needs analysis be conducted in a systematic manner?

**Think about each of the 'WH' questions: Who, what, when, where, why, how?**

- Who are the stakeholders and what are their requirements?
- What items conflict, and there any compromises?
- What are the advantages and limitations?
- What are the inputs to the system, what are the expected outputs? (Very important)

# Systems Design Considerations - Needs Analysis - Mobile Phone Example

Stakeholders and their high-level needs:

**Consumers**

- Connectivity (Wifi, Bluetooth, USB, Network)
- Aesthetic Appearance (size,weight,shape)
- User-friendly experience - Functionality

**Supplychain and Sales**

- Marketing - A point of Difference
- Volume Production - Achieve economy of scale
- Profitability

These can be related to one another, and broken down into more specific requirements.

# Top-Down Segmentation Approach - Mobile Phone Example

A Top-Down Segmentation Approach to design involves breaking a system into comprehensible elements and solving problems individually. Of course there is overlap between the different system components.

This can be studied by reverse-engineering the process Eg. Breakdown of a mobile phone into mechanical, electrical and software expectations:

- Outer Case
- Display
- Input Capture (Buttons / touch screen)
- Printed Circuit Board
- Operating System and Apps

# Top-Down Segmentation Approach - Mobile Phone Example

Each of the systems mentioned on the previous slide can be further broken down into seperate components. Eg The PCB may consist of the following:

- Microcontroller / microprocessor
- Audio and Visual Driver ICs
- Memory
- Buttons
- GPS Module
- SIM Card Identification
- Radio Antenna
- Bluetooth Module
- Lights

# Top-Down Segmentation Approach - Mobile Phone Example

Again, The individual modules on the PCB can be broken down into more elementary components.

Keep asking "What does this aspect of the system consist of?" The process is similar to that of root-cause analysis where it is instead asked "Why does this issue exist?".

Due to the integrated nature of systems (and indeed Mechatronic solutions) challenges arise where design aspects have dependency on other components. This will be addressed shortly in System Architecture Diagrams.

# Design for Contingency / Future Proofing

## What needs to be considered during design for contingency / future proofing?

What happens to the system if one component fails / stops working? Can the system overcome this?

At a minimum, can it fail gracefully? eg. automatically switch to alternative power source

Can new features / components be added to the system?

Are there chips that offer "drop-in-replacement"? eg. if current capacity has to be increased

# Design for Contingency / Future Proofing - Examples


Branch out superfluous communication transceivers - they may offer additional functionality or facilitate new features.

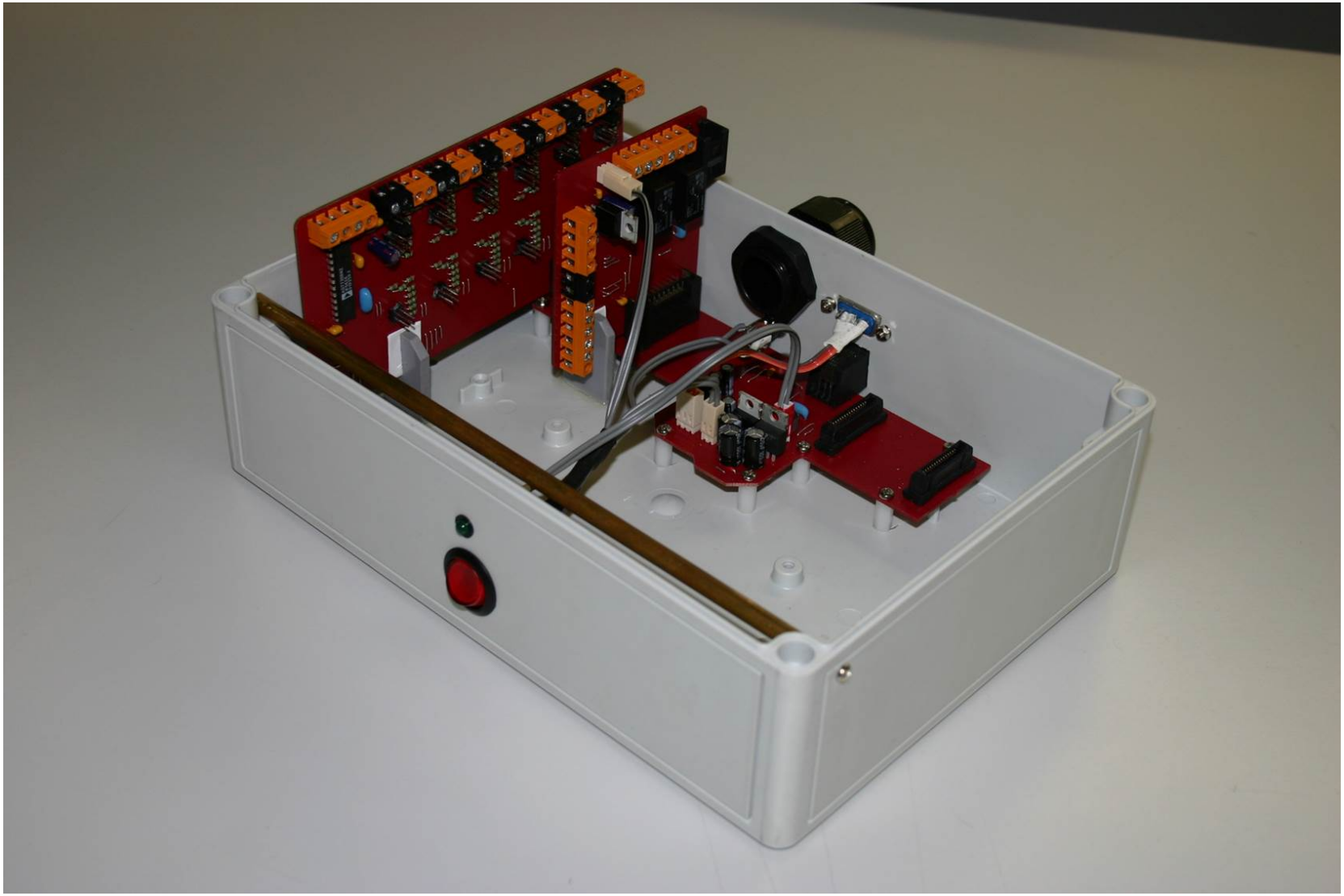
During prototyping, and even in production runs, branch out test pads. They'll be invaluable when troubleshooting.

If using addressing methods, leave some addresses free.

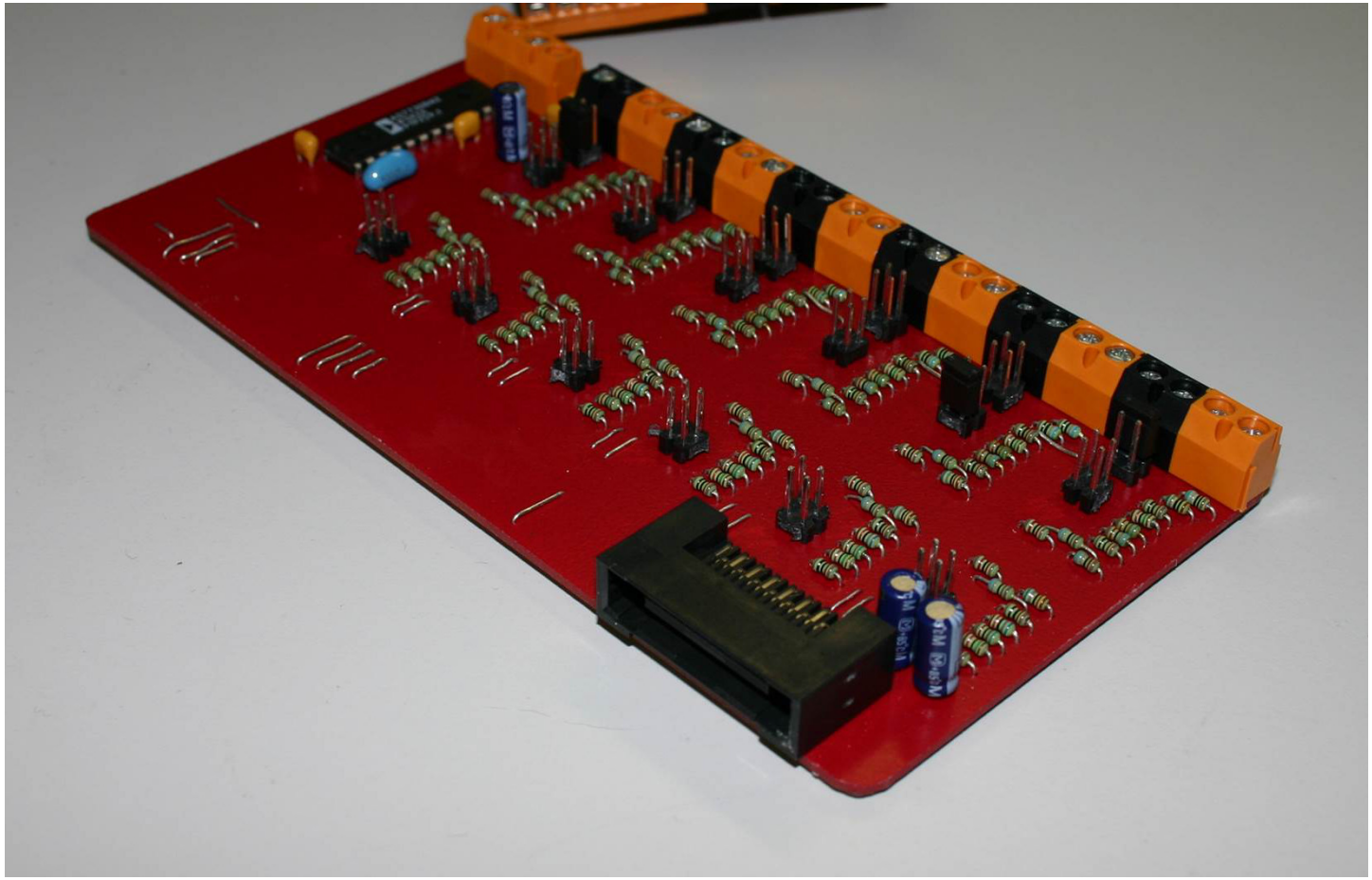
Consider, will the system ever have to synchonise with another? Will a digital trigger be required?

Follow industry standard layouts.

# Design for Contingency / Future Proofing



# Design for Contingency / Future Proofing



12 Channels Selectable Resistive, Transistive, or Digital Sensors, Self Powered or Driven, Selective voltage division, Frequency tunable analog filter

# Design for Modularity / Scalability

## How can a system be made modular / scalable?

Can the system be modularised? eg. a Lego format?

What is the size of the address space? Can it be expanded? Remember 32-bit IP addresses were once thought to be enough.

What are the limits for scalability? is it track length, communications noise, current consumption?

Can the system be assembled in a modular manner? Are there any dependencies?

Are the components servicable? Can they be maintained?

# Design for Modularity / Scalability - Examples

Cusby - A modular USB to peripheral adapter.

Scaling horizontally (Adding more nodes to a system) eg. Multiple processors / Modules / (like cusby)

Scaling vertically (Adding resources to a single node in a system) eg. Making Arduino shields

Modularity can help with decentralisation and failure mitigation

# Design for Manufacture

## How can systems be designed for manufacture?

How will the system (eg. a PCB) be assembled, and mounted into the final product?

Which tools need access during the manufacturing process?

Will all of the connectors be easily reachable?

How can the components be held? Are there any restricted areas?

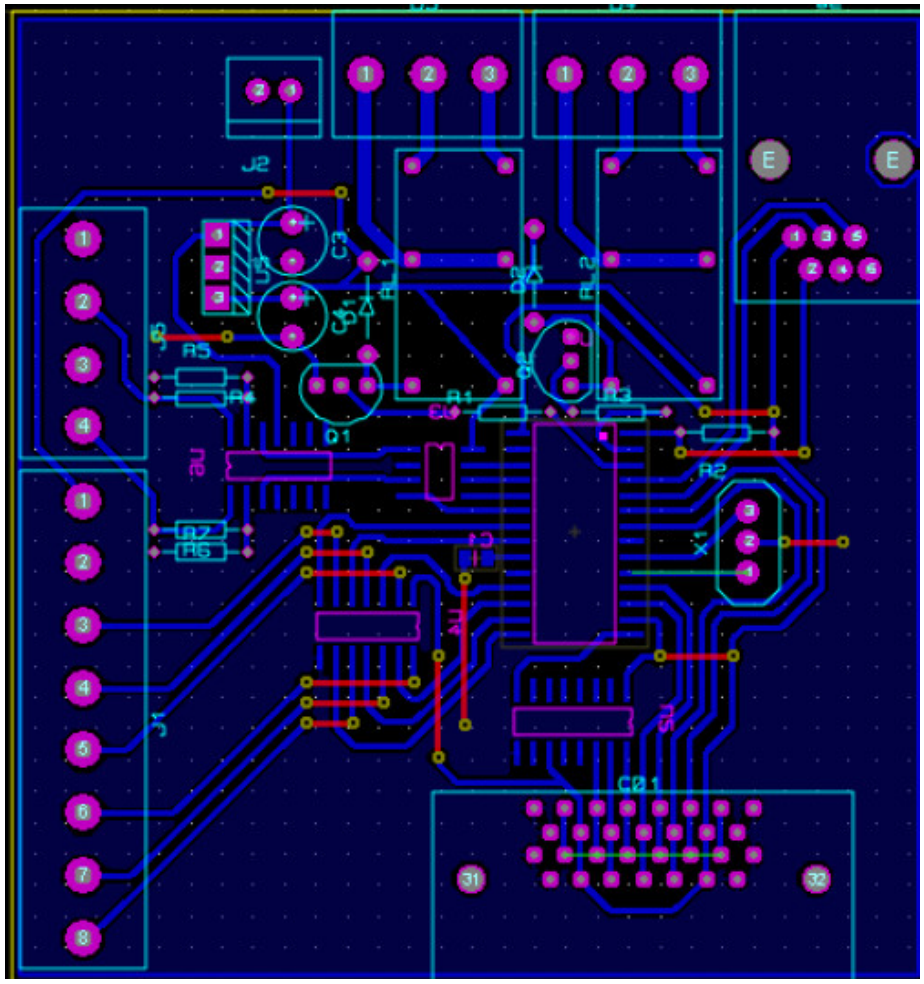
# Design for Manufacture - Examples

Mark orientations clearly (eg. pin 1 of a chip)

Ensure that components can only be assembled in one way (eg. can't be plugged in backwards / upside down) unless they work both ways.

Consider the order of assembly operations. Are there any sub-assemblies that can be manufactured in parallel?

# Design for Manufacture




Chip layouts and orientation of components with rotational symmetry are identified

# System Architecture Diagrams

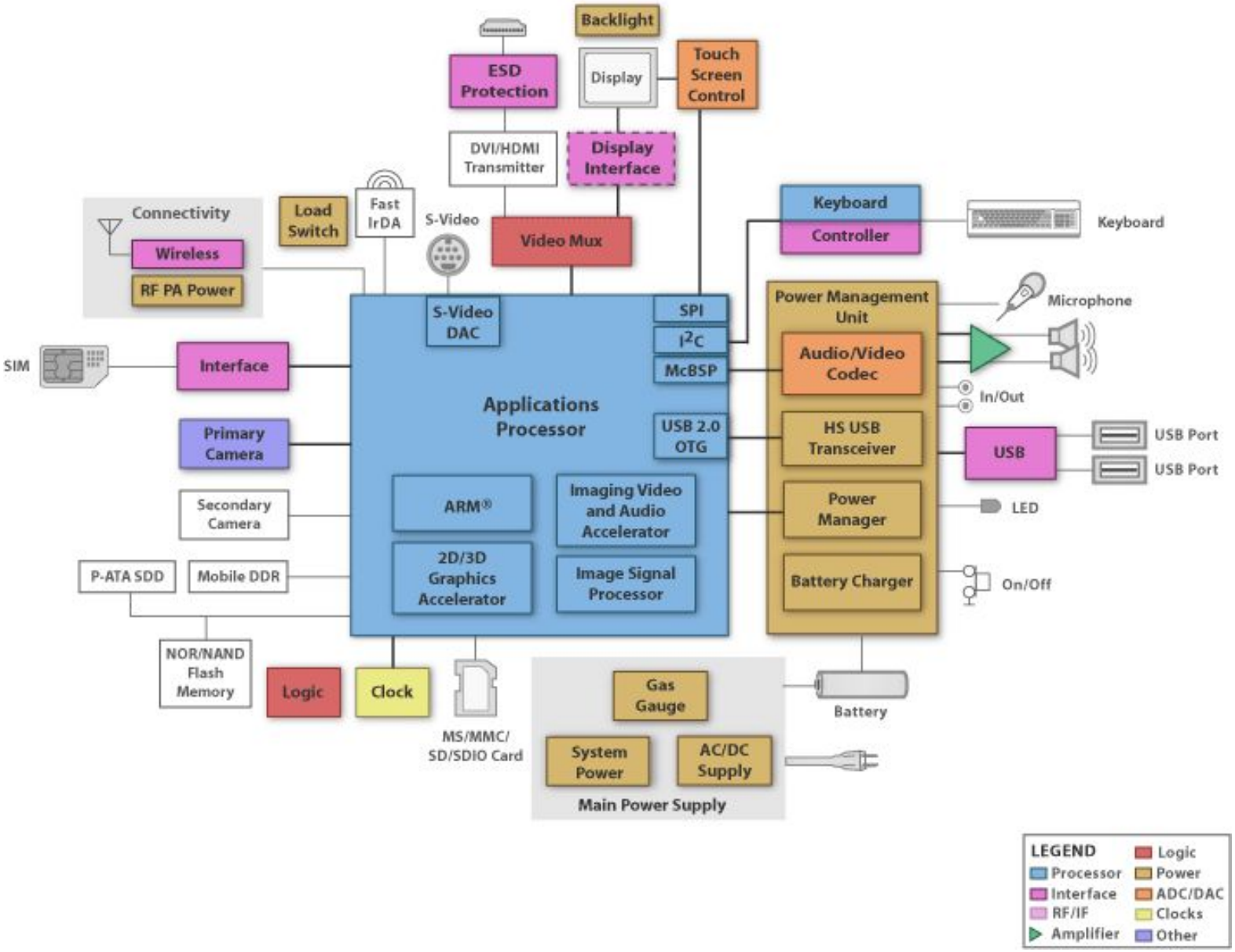
System Architecture Diagrams are a useful tool for structuring a design process and to look at high-level interactions and dependencies of system components.

The nodes represent system components, which are connected by different interfaces. In electronics, this is typically analog, digital, or wireless communication / information protocols.

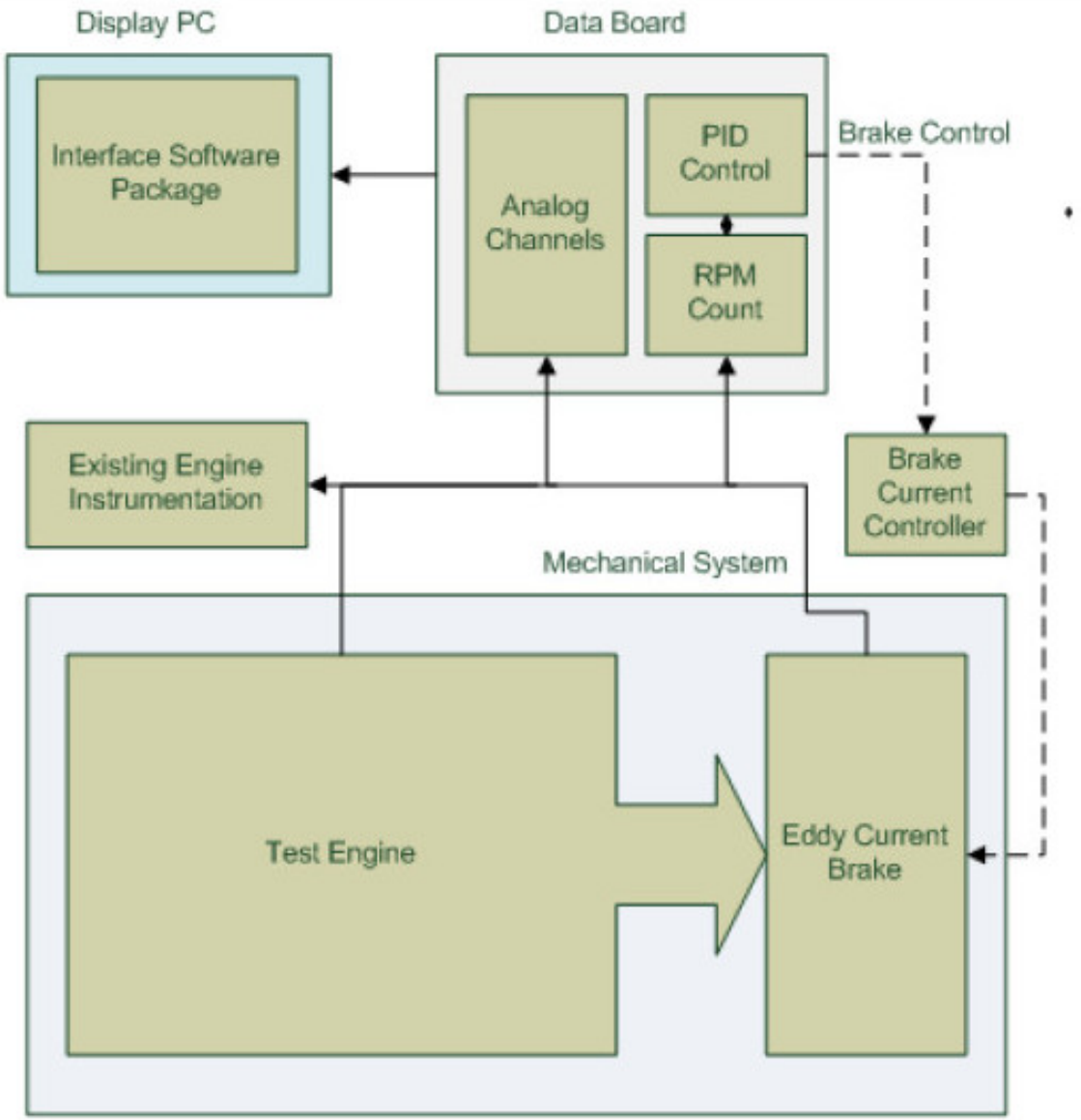
## System Architecture Diagram

- Nodes should be labelled with their functionality
- The connections should be labelled by the type of expected interface.

# System Architecture Diagrams

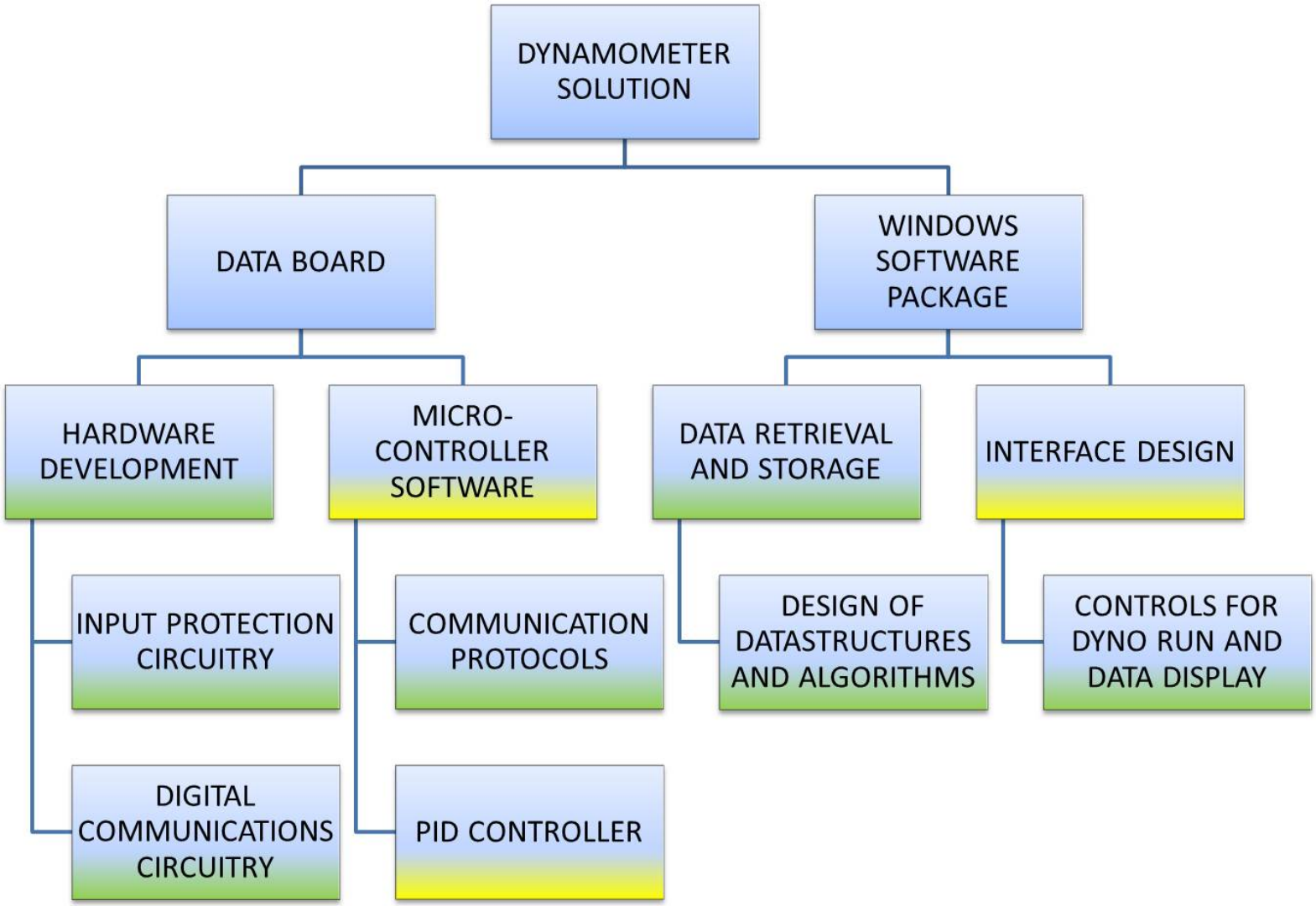


# System Architecture Diagrams



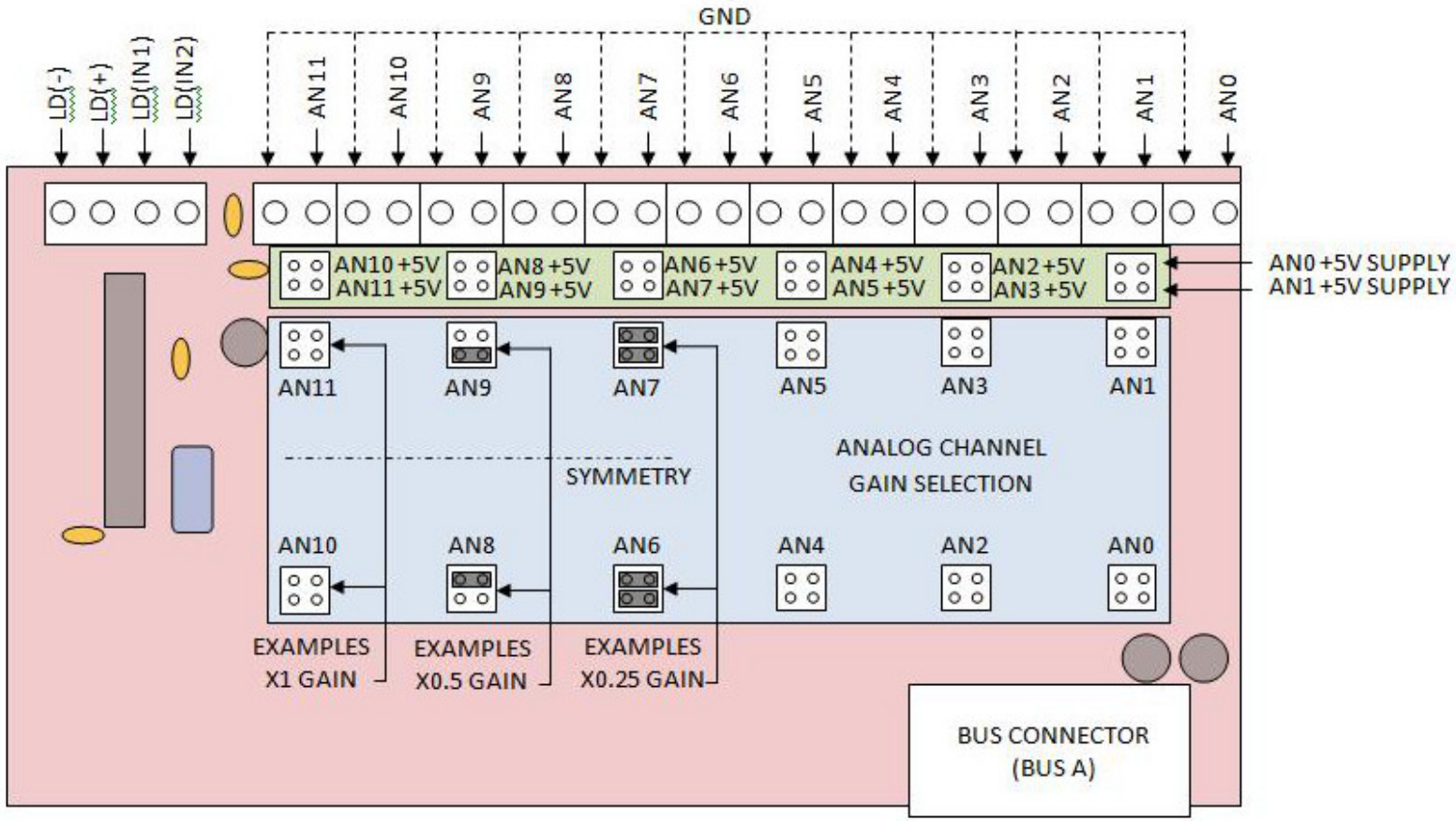
# Relationship and Dependency Diagrams

Hierarchy showing how the overall system has dependency on child (elementary) components

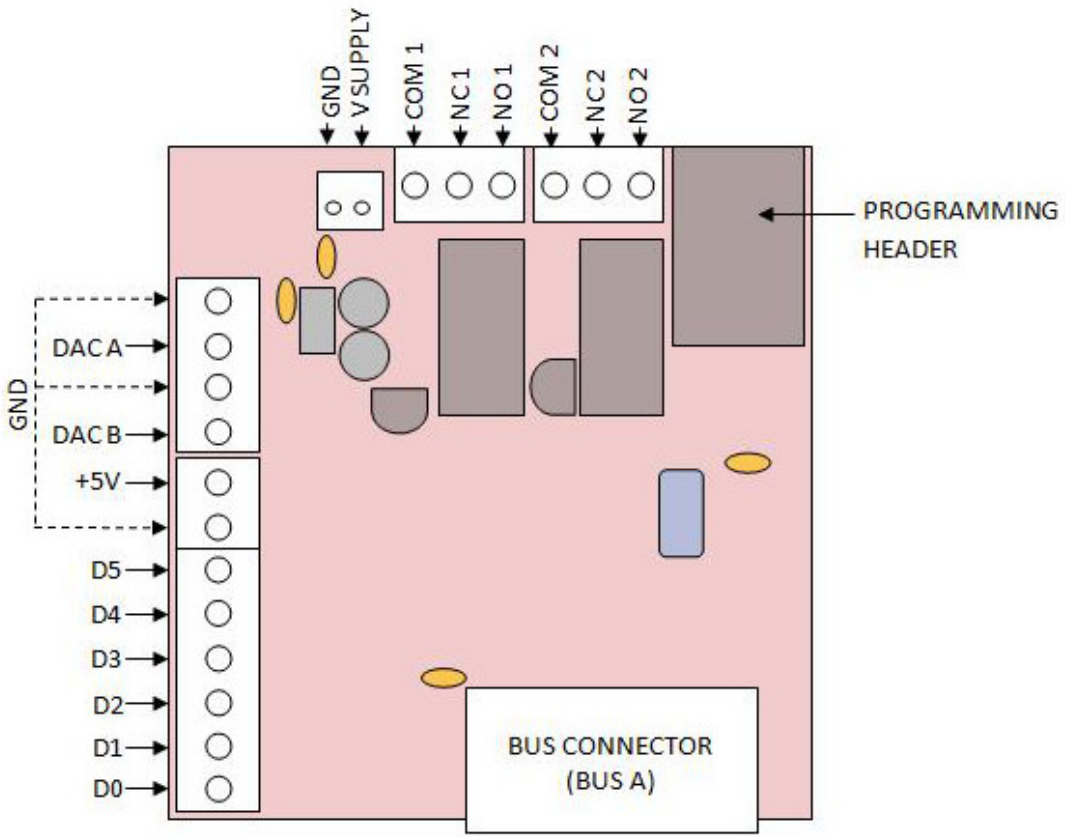


# Design Documentation

Different levels may be required to show how the system is assembled and works. Eg. Wiring Diagrams



# Design Documentation



# Communication Protocols

There are a range of communication methods for electronic systems, each of which have their own distinct advantages and disadvantages.

## Self-Reflection on Communication Selection

- Which communication methods have you used?
- What are your current selection methods?
- Have you ever been caught out by your choices?
- What makes one type of communication more suitable than another?

# Communication Protocols - Considerations

## Communication Protocol Selection Considerations

- What is the communication topology? Eg. one-to-one or broadcast? Is it a star or daisychain topology?
- Is the communication synchronous or asynchronous?
- Is there a dominant processor in the system?
- What data rate is required?
- How far must the information travel?
- What processing overhead is required?
- How does each unit know if/when/how it can "speak"?

# Communication Protocols - Bus Specifications

| Name | Topology | Addr. | Bitrate | Wires | Length |
|---|---|---|---|---|---|
| RS232 | 1:1 | Wired | 1 Mbps | 3/5 | 15 m |
| RS485 | 1:Few | Soft | 35 Mbps | 3 | 1.2 km |
| USB | 1:1 | Soft | 480 Mbps | 5(P) | 5 m |
| CAN | 1:Many | Soft | 1 Mbps | 4(P) | 40 m |
| $I^2C$ | 1:Many | Soft | 3.4 Mbps | 2 | 30 cm |
| SPI | 1:Many | Hard | > 10Mbps | 3+(A) | 30 cm |

(P) - Includes power
(A) - Have to add lines for each address, or use an addressing method

# Communication Protocols - Advantages and Limitations

Each communication method has advantages and limitations.

For onboard, or intermodular, communication I2C or SPI are the obvious contenders. I2C requires fewer wires, and works in a software addressed daisychain manner, whereas SPI has hardware chip selection which requires more tracks, but offers faster communication.

Uarts on microcontrollers readily convert to RS232 or RS485 through transcievers. RS232 was historically available on PCs. USB communication has largely replaced this functionality and is now available natively on some microcontrollers.

# What to take away from this lecture

## The Main Points

- Systems Design begins with analysing the needs of stakeholders and converting them into a specification
- Systems Design involves consideration of the whole product life cycle including: assembly / interfaces / manufacturing / serviceability / expansion / future-proofing
- System Architecture Diagrams and Relationship / Dependency Diagrams can be used as a tool to look at how parts of the system relate to one another, and what happens in a failure.
- System Architecture Diagrams can also be used to select an appropriate communication protocol.