Arduino Programming
Part 7: Flow charts and Top-down design

EAS 199B, Winter 2010

Gerald Recktenwald
Portland State University
gerry@me.pdx.edu
Goals

Introduce flow charts
- A tool for developing algorithms
- A tool for documenting algorithms
- A visual method of communicating about any sequential or iterative process
- Great for visual learners!

Top-down design
- One technique for creating a plan for large, multi-step problems
- Not tied to flow charts, but can be used effectively with flow charts
Flow chart symbols

Terminator: Start or stop a sequence. May contain module name.

Process: A step in the process or computational algorithm.

Data input: Information from outside of the algorithm or process.

Decision: Choose a flow path for continuing the algorithm or process.

Flow indicators: Connect other elements.

Connector or Junction: Optional joint where flow indicators merge.
Exercise 1

Draw the flow chart to read and display the salinity value on the LCD monitor

Keep it simple

❖ 5 or so symbols (not counting arrows)
❖ Describe only the high level actions
Exercise 1

Your answer goes here.
Exercise 1

1. Read and display salinity
   - Specify constants
   - Initialize LCD
   - Read salinity
   - Display value to LCD
Exercise 2

Expand the “Read salinity” step in another flow chart

- Keep it simple
- “analog data” is an external input

![Flowchart diagram](image_url)
Exercise 2

Your answer goes here.
Exercise 2

1. Read salinity
2. Turn on power
3. Wait
4. Read analog input
5. Turn off power
6. Stop

- Output pin, input pin
- Analog value
Exercise 3

Expand the “Read analog input” step in another flow chart

- Compute the average of n readings
- “analog data” is an external input
Exercise 3

Your answer goes here.
Exercise 3

Read average analog input

n readings, input pin

Initialize: sum=0, counter=0

Read analog input

Add to sum, increment counter

Counter<n?

yes

no

Average = sum/n

Stop
Top-down design

1. Start with a general statement of the solution
   a. List the main steps
   b. Don’t worry yet about details

2. Pick one of the steps
   a. Break this step into a manageable number of sub-steps
   b. Don’t worry about too many of the details
   c. Apply step 2 to one of steps just generated
Top-down design

Recursive refinement: from general to specific

Repeat refinement until individual steps can be translated into concrete actions or lines of code.