E11 - Autonomous Vehicles

Lecture 17 – Debugging
Debugging Outline

- General Approach
- Tools
- Parting Words
1. Read the Code
2. Establish Feedback
3. Determine expected outputs
4. Begin your search
1. **Read the code**

   - Read the code at the high level function calls to allow understanding of the architecture and control flow
     
     **E.g.** `void loop() {...}

   - Read the individual function calls
     
     **E.g.** `void getMax(int a, int b) {...}`
2. Establish Feedback Mechanism

- In Arduino, we have
  ```
  Serial.Print(The value is);
  Serial.Println(x);
  ```

  Note: `Println()`’s are not real time and do not always indicate how far along in the code you have proceeded.

  Note: In many languages there are also debugging tools like breakpoints, watchdogs, …
3. Determine Expected Outputs

- At key INTERMEDIATE points in the program, understand what values you SHOULD be getting.

E.g.

```java
int t = getTemperatureInC(double tInF);
```
4. **Search**

- What are we searching for?
  
  Bug 1: Wrong Output
  Bug 2: Crashes, gets stuck
4. **Search** – Wrong Output

- Must systematically check the values of all outputs leading up to the wrong output
- The system could be:
  - Check output on line 1
  - Check output on line 2
  - Check output on line 3
  ... 
- Often we use a bisection search!
4. **Search** – Program Crashes/Hangs

- Must systematically eliminate code that could cause the problem
- The system could be:
  - Comment out all but line 1
  - Comment out all but line 2
  - Comment out all but line 3
  ...
- Often we use a bisection search!
4. **Search** – Bisection search

Does it work up to half way?

YES!
4. **Search** – Bisection search

Does it work up to 3/4 way?

NO!
4. **Search** – Bisection search

Does it work up to here?
4. **Search – Bisection search**

```cpp
void setup()
{
  Serial.begin(9600);
  Serial.print("\n");
  Serial.print("\n");
  int smallest1;
  if (1 < 2)
    
## smallest1 = 1;
  else
    
## smallest1 = 2;
  
int smallest2;
  if (10 > 20)
    
## smallest2 = 10;
  else
    
## smallest2 = 20;
  
int smallest;
  while (smallest1 < smallest2)
    
## smallest = smallest2;
  
## smallest = smallest1;
  Serial.print("The smallest of 1 and 2 our two values is ");
  Serial.println(smallest);
  Serial.print("\n");
  Serial.print("The smallest of 10 and 20 our two values is ");
  Serial.println(smallest2);
  Serial.print("\n");
  Serial.print("The smallest of 1, 2, 10, 20 is ");
  Serial.println(smallest);
}
Debugging Outline

- General Approach
- Tools
- Parting Words
Tools

- Modularize
- Backup Systems
- Debugging Hardware
- Work with others
- Walk away…
Tools

- Modularize
  - Divide and conquer your code!
  - Each module (e.g. function) can be debugged individually.
Tools

- Backup Systems
  - Make sure you back up working Modules often
  - There exist software that does “Version Control” for you.
Tools

- **Debugging Hardware**
  - SIMULATE FIRST!!!!!!
  - Start with optimal values, the easy occurrences. Then increase the complexity.
  - If your code won’t work with an “ideal” sensor reading, it won’t work with the real stuff.
  - If your code works perfectly with ideal measurements, it “may” be the hardware causing a problem.
Tools

- **Work with others**
  - Explaining the problem can solve the problem
  - Online resources are invaluable
  - Solving other people’s problems will help you in the long run
  - Paired Programming
Debugging Outline

- General Approach
- Tools
- Parting Words
Parting Words

- Simple things
  - Check the power source!
  - Is the switch on?

- Practice! Practice! Practice!
Approaches to Control

- Read the code
- General Approach
- What is your feedback? PrintIn()? 
- Isolate the bug
- Bisection Search
- Modularize
- Explain to a friend
- Backup your code!
- Check the Plug
- Hardware vs Software
- If you didn’t fix it, its not fixed.
- Reproducable bugs