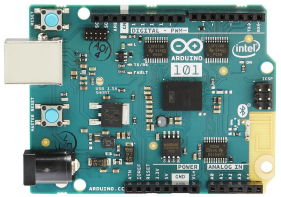
Bifröst

Visualizing and Checking Behavior of Embedded Systems across Hardware and Software

Will McGrath – wmcgrath@stanford.edu
Jeremy Warner – jeremy.warner@berkeley.edu

e**M**bedded **G**ateway **C**loud [MGC]

embedded

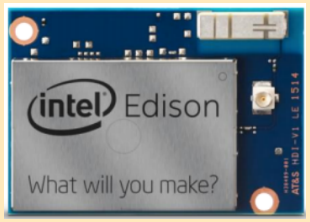


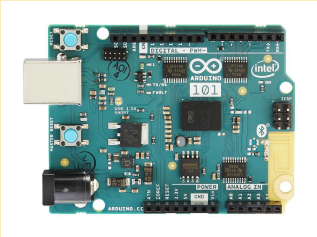
gateway

cloud

e**M**bedded **G**ateway **C**loud [MGC]

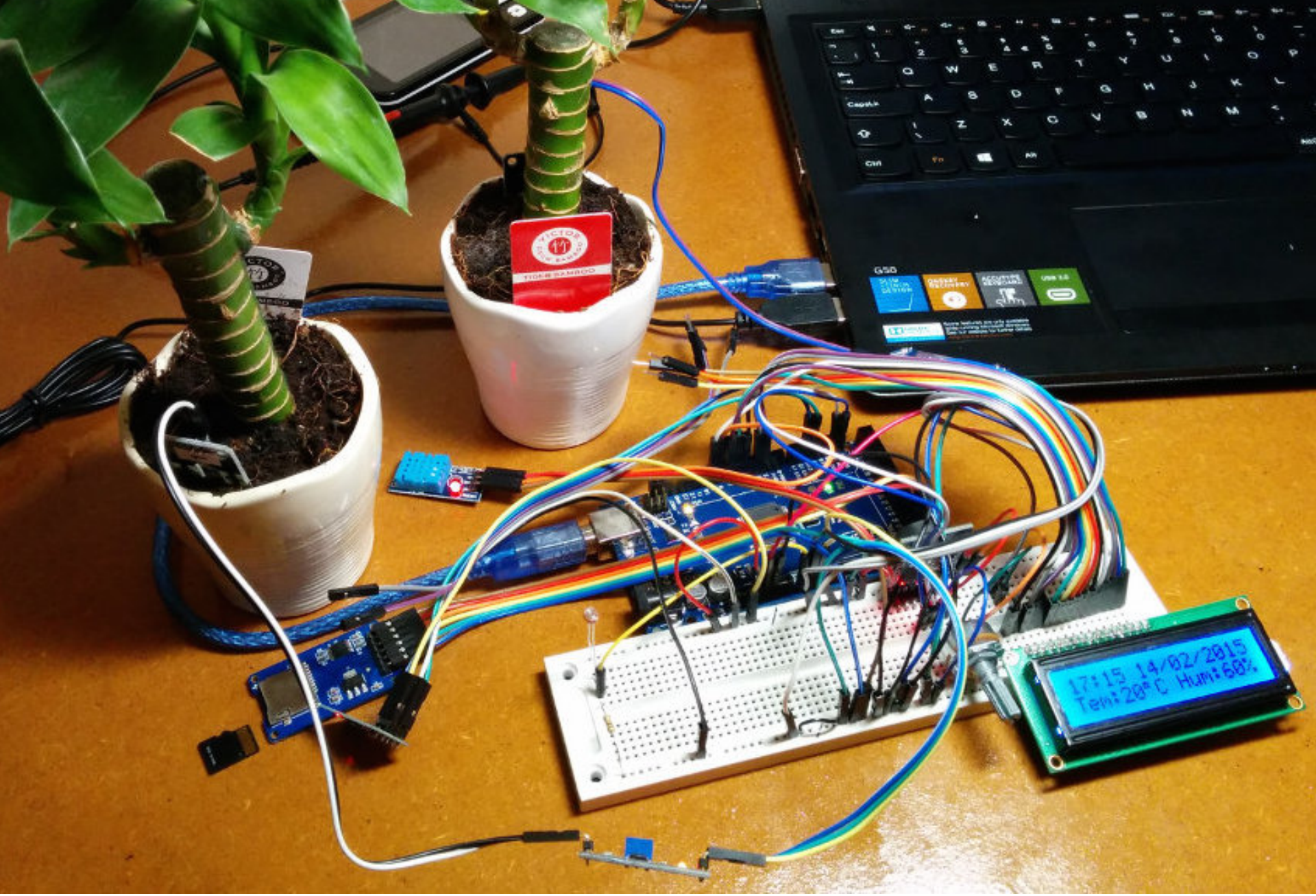
embedded

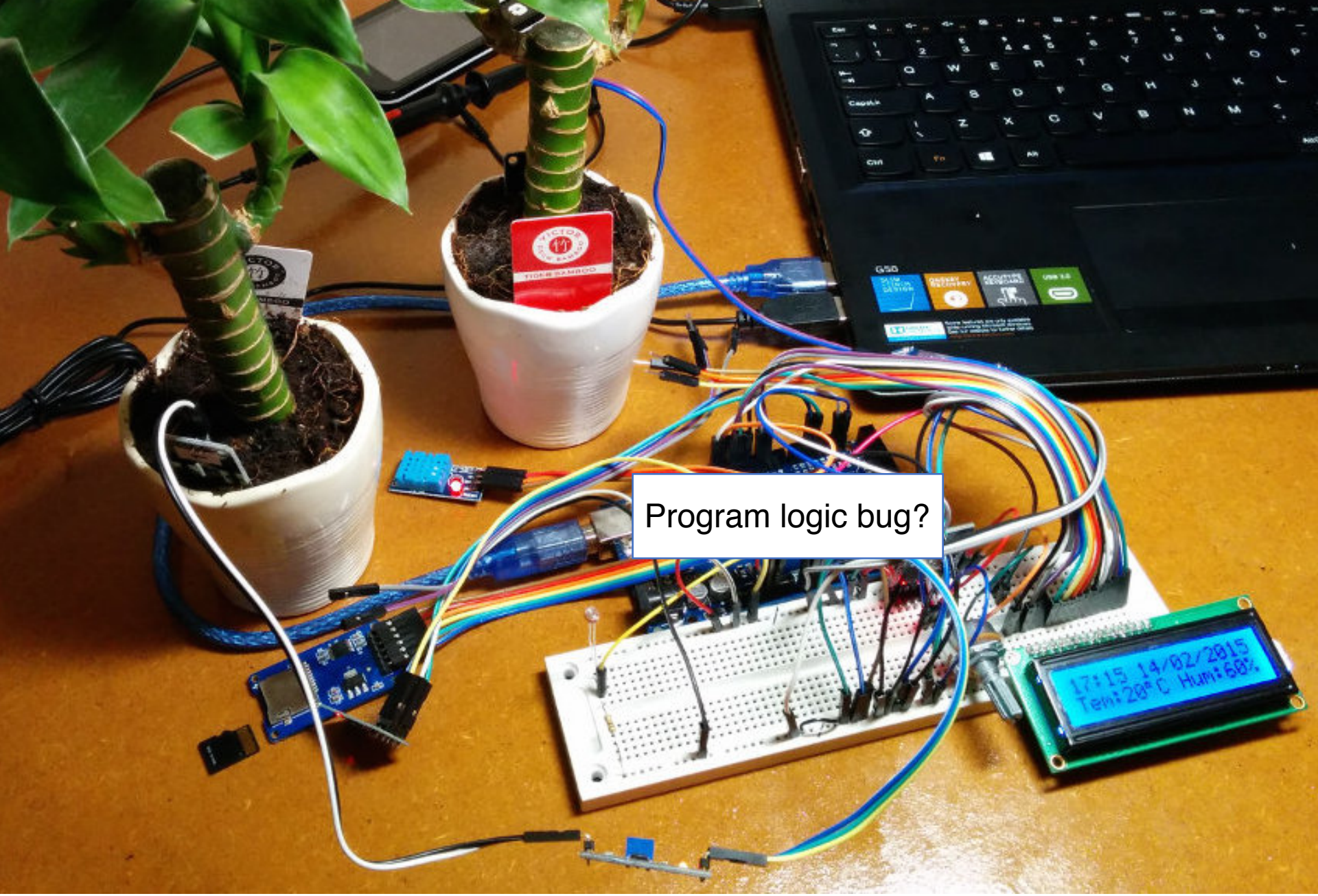




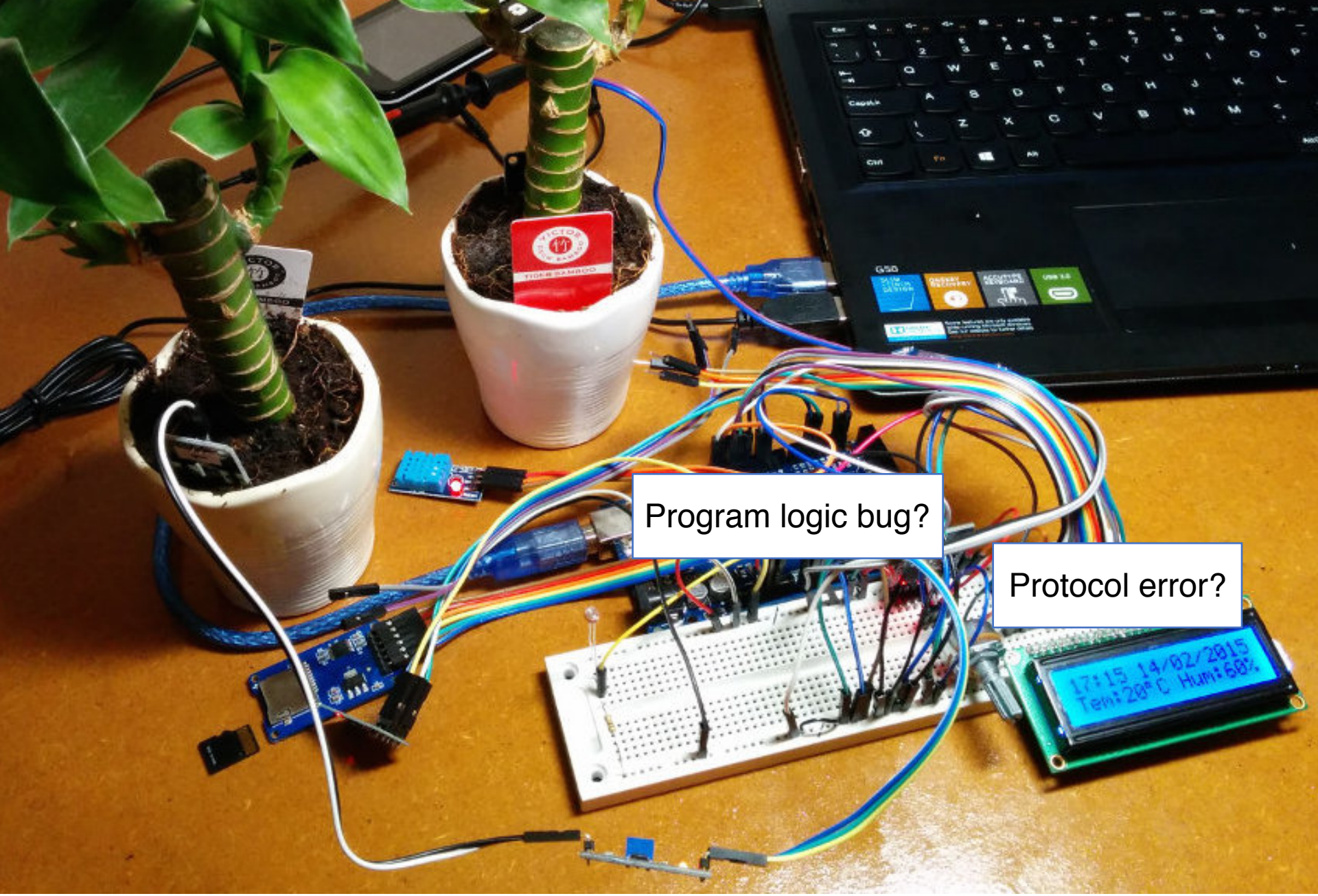
gateway

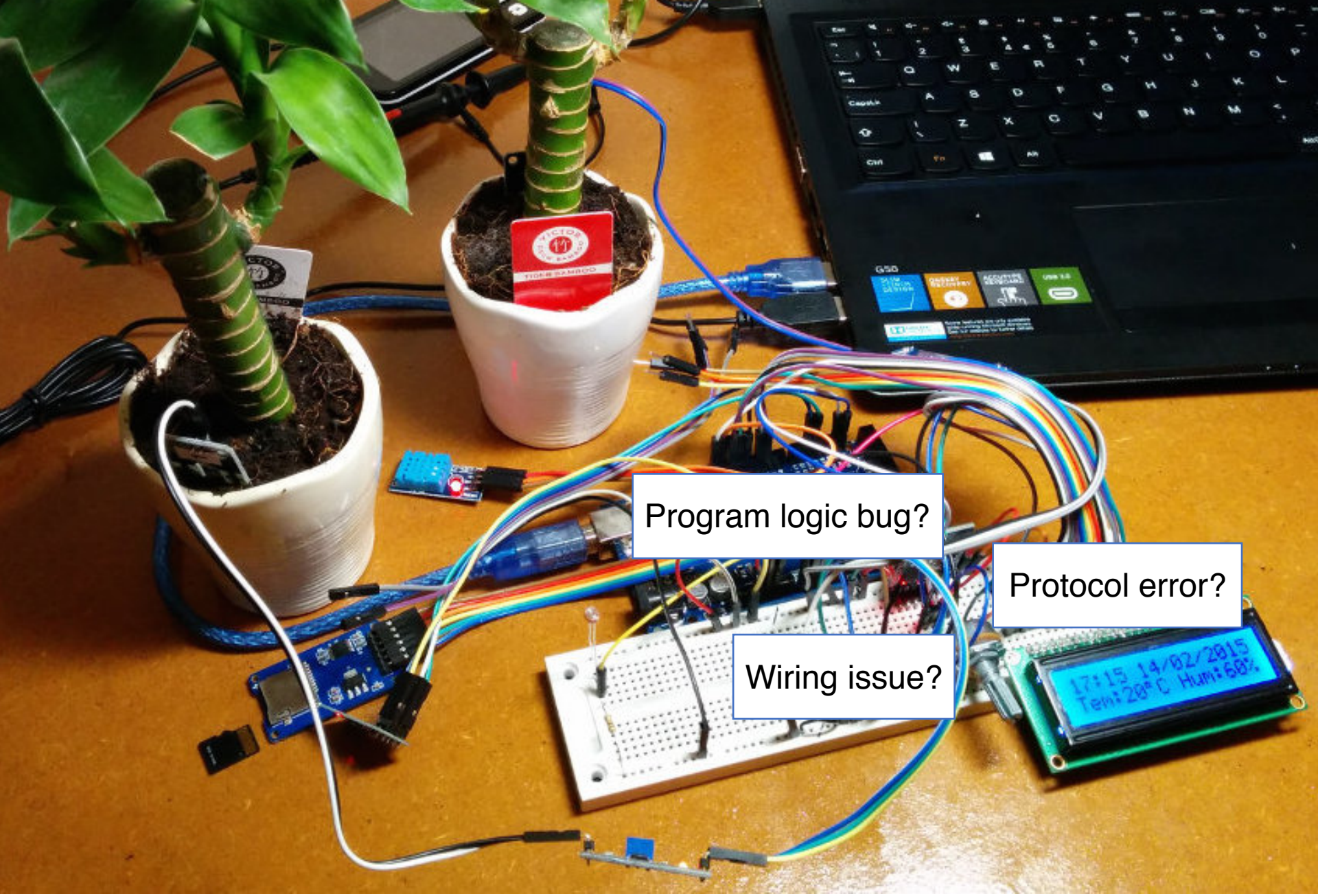
cloud

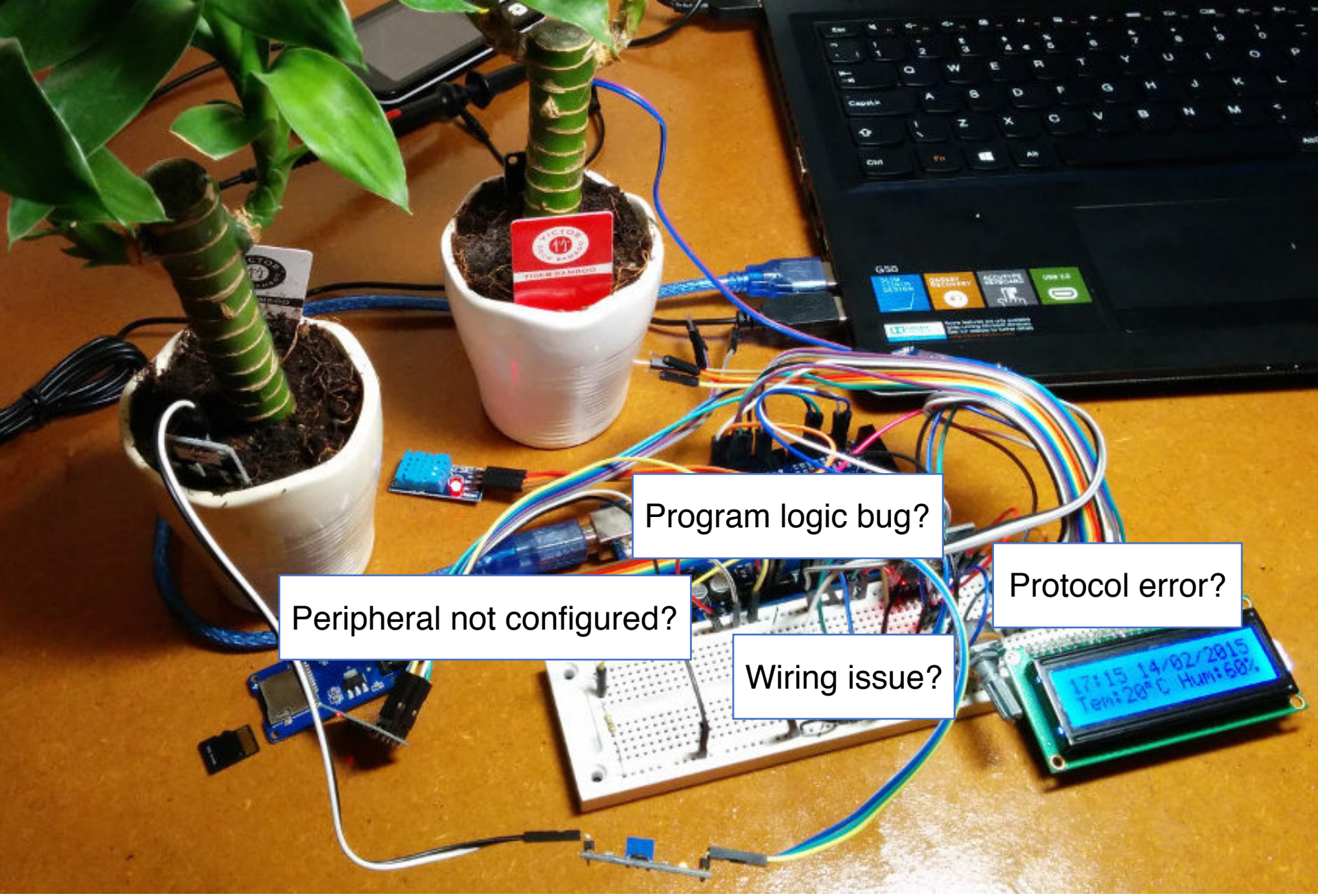

17:15 14/02/2015
Tem:20°C Hum:60%

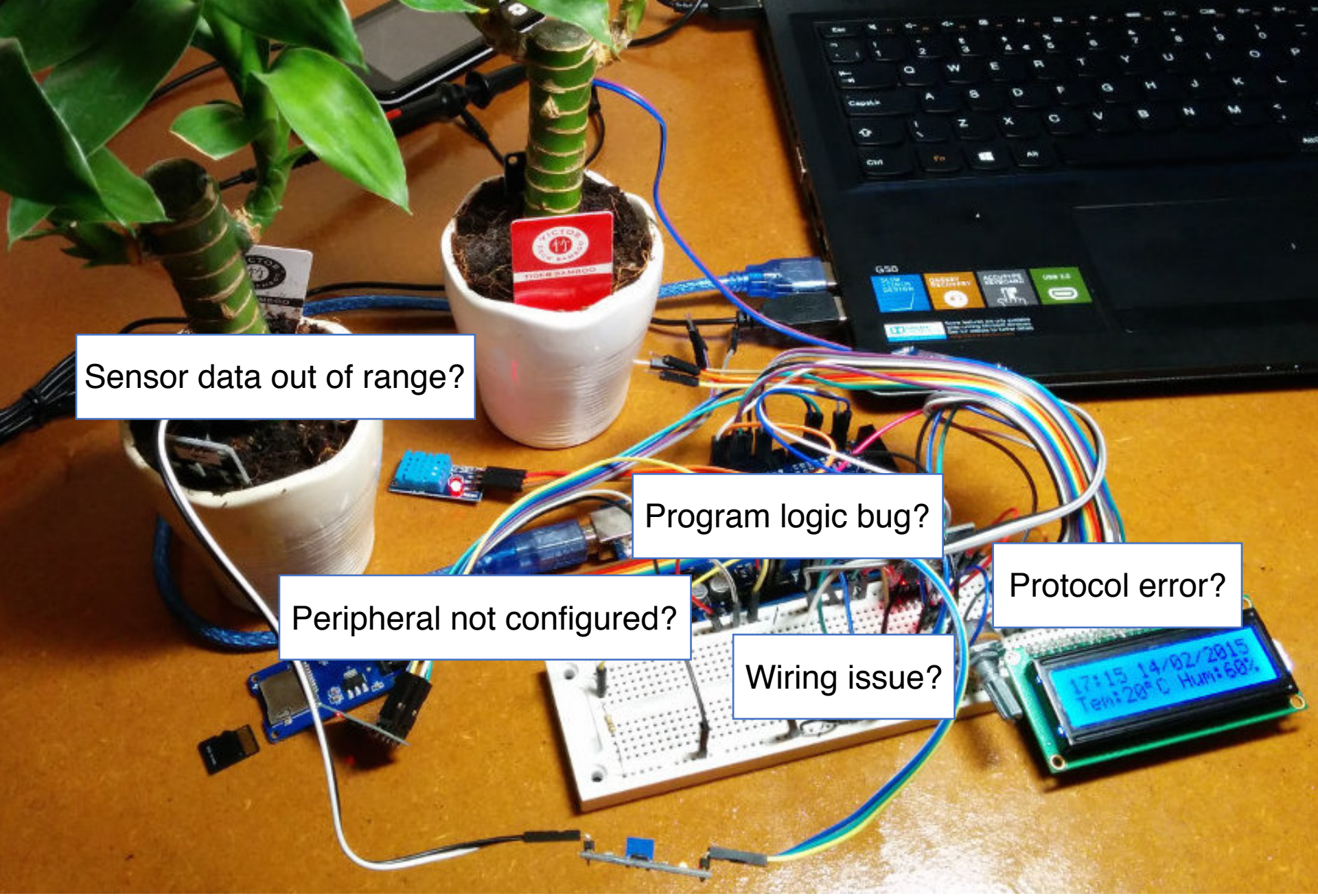


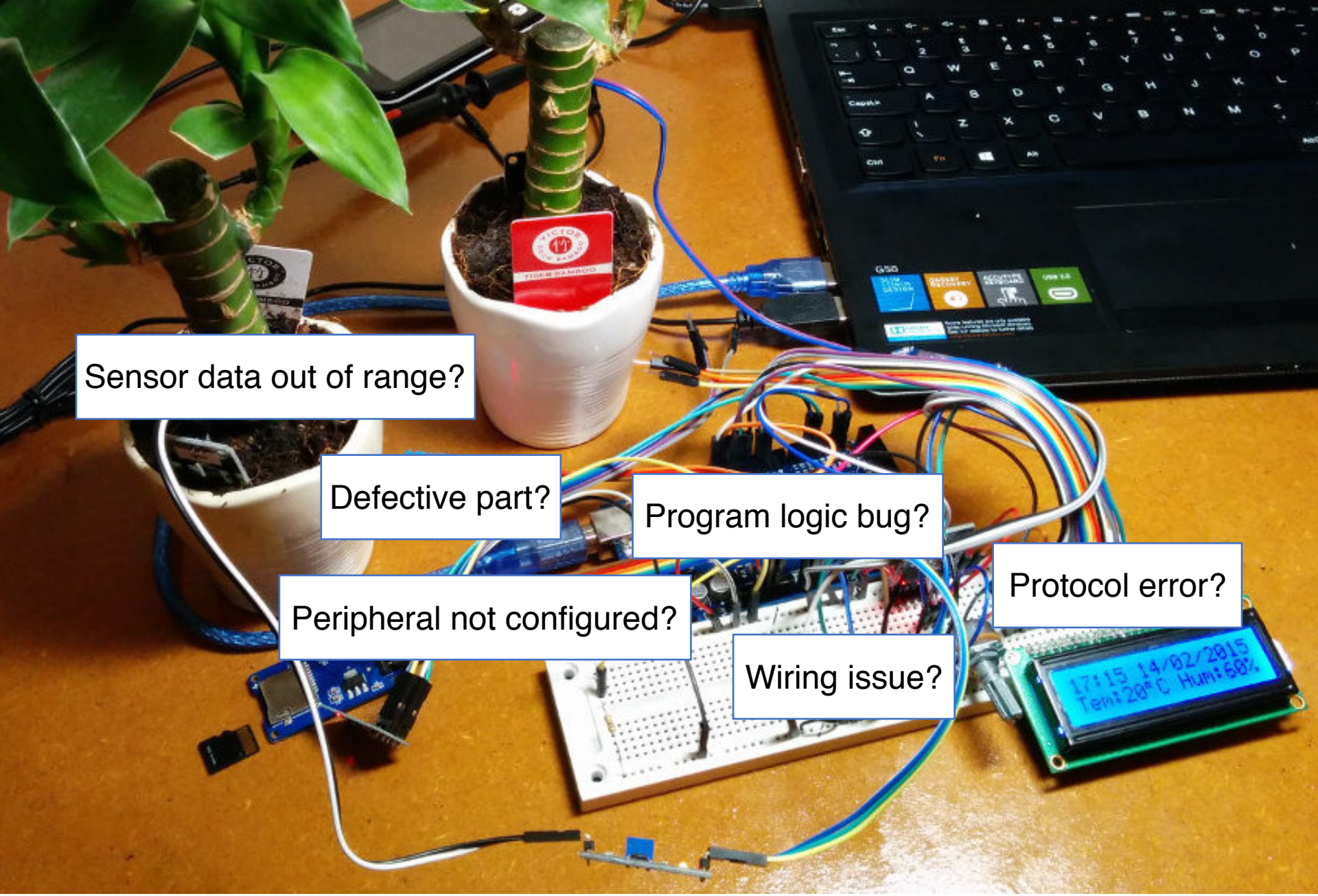
Program logic bug?

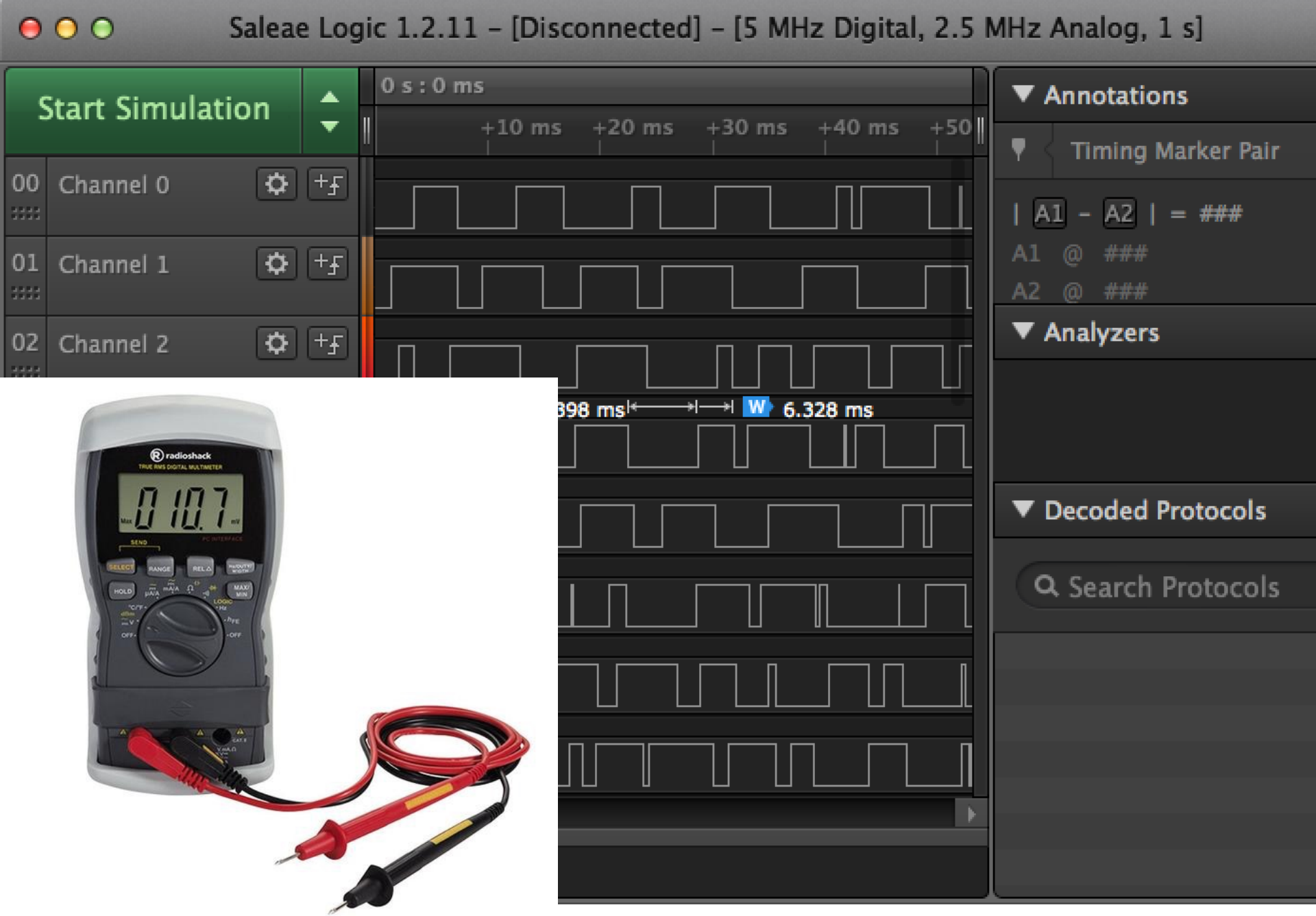

Program logic bug?
Protocol error?
17:15 14/02/2015
Tem:20°C Hum:60%

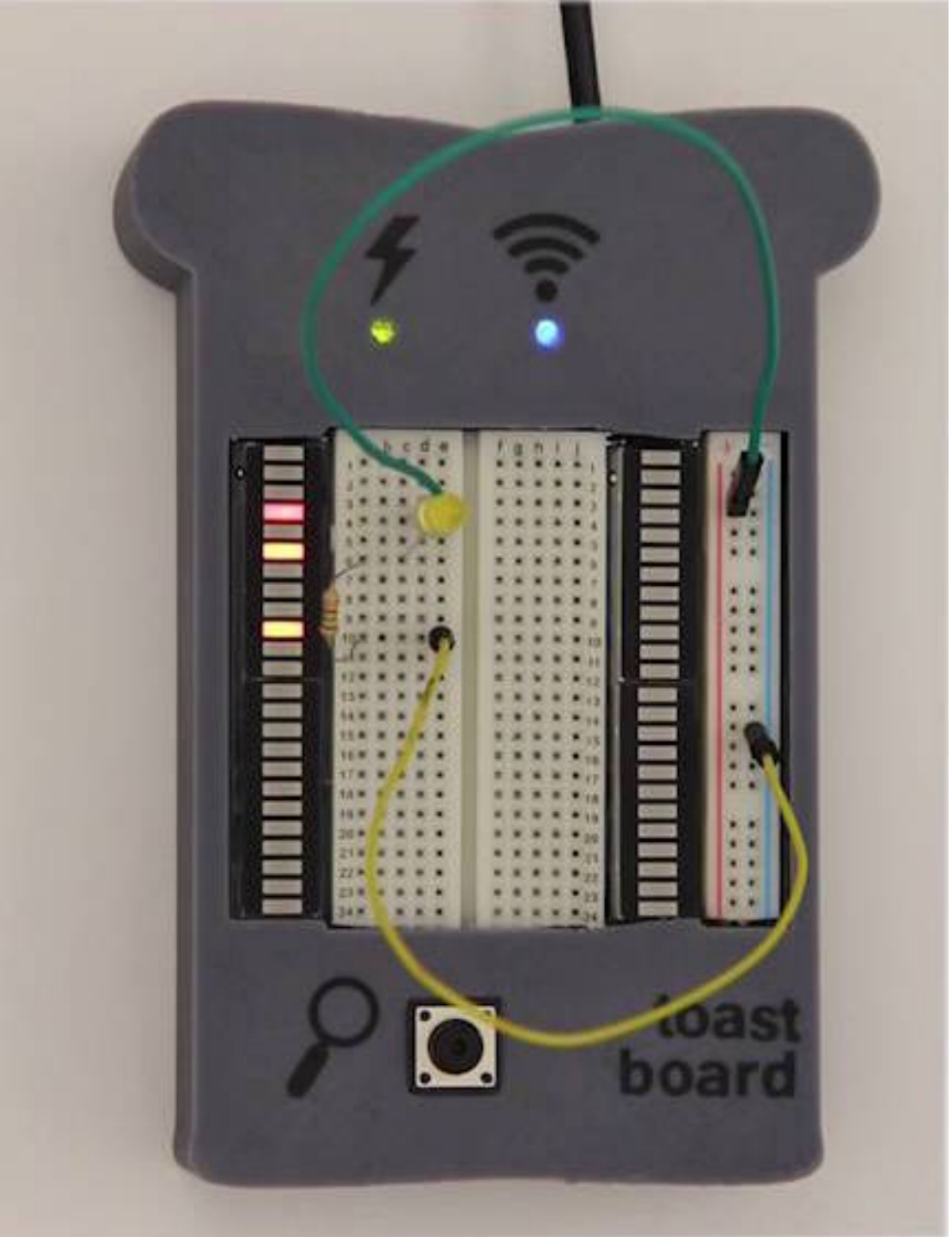

Program logic bug?
Protocol error?
Wiring issue?
17:15 14/02/2015
Tem:20°C Hum:60%


Program logic bug?
Protocol error?
Peripheral not configured?
Wiring issue?
17:15 14/02/2015
Tem:20°C Hum:60%


Sensor data out of range?
Program logic bug?
Protocol error?
Peripheral not configured?
Wiring issue?
17:15 14/02/2015
Tem:20°C Hum:60%


Sensor data out of range?
Defective part?
Program logic bug?
Protocol error?
Peripheral not configured?
Wiring issue?
17:15 14/02/2015
Tem:20°C Hum:60%


Saleae Logic 1.2.11 – [Disconnected] – [5 MHz Digital, 2.5 MHz Analog, 1 s]
Start Simulation
0 s : 0 ms
+10 ms
+20 ms
+30 ms
+40 ms
+50
00 Channel 0
01 Channel 1
02 Channel 2
398 ms
W
6.328 ms
Annotations
Timing Marker Pair
| A1 - A2 | = ###
A1 @ ###
A2 @ ###
Analyzers
Decoded Protocols
Search Protocols
radioshack
TRUE RMS DIGITAL MULTIMETER
SELECT
RANGE
REL Δ
HOLD
MAX/MIN



Toast Board

Measures analog voltages at all breadboard rows

Compares measured results to intended circuit to identify errors

No embedded software support

Button §

```
// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
    Serial.println("LED ON!");
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
    Serial.println("LED OFF!");
  }
}
```

Button §

```
// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
    Serial.println("LED ON!");
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
    Serial.println("LED OFF!");
  }
}
```

wmcgrath — arm-none-eabi-gd — 84×24

openocd.bin ... arm-none-eabi-gd

```
William-McGraths-MacBook-Pro:~ wmcgrath$ salamigdb
GNU gdb (GNU Tools for ARM Embedded Processors) 7.6.0.20140228-cvs
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=x86_64-apple-darwin10 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
0x00000858 in ?? ()
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x81000000 pc: 0x00000858 msp: 0x200023a0
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x81000000 pc: 0x00000858 msp: 0x200023a0
(gdb) b modfile.ino:27
Breakpoint 1 at 0x2164: file modfile.ino, line 27.
(gdb)
```

Existing Embedded Debugging Techniques

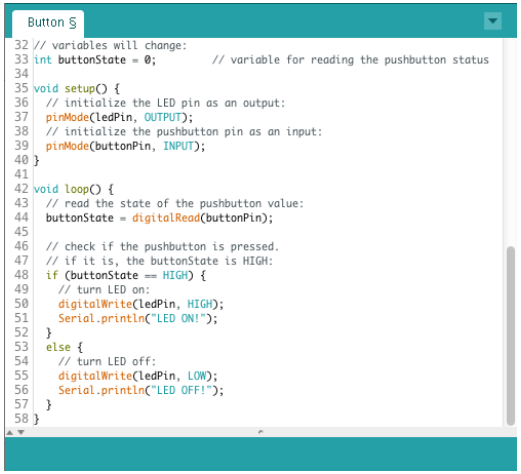
- Multimeter
  - Point to point
  - Obscures rapid changes
- Oscilloscope/Logic Analyzer
  - Fast, but expensive
  - Nontrivial configuration
- Print Debugging
  - Easy, but weak time information
- GDB
  - Powerful, but not always available
  - Complex configuration

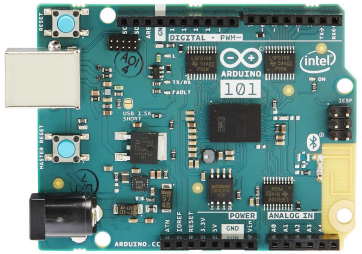
In contrast with prior work which has focused on providing tools only for debugging software *or* circuits,

we directly address the **interconnected nature** of embedded systems projects.

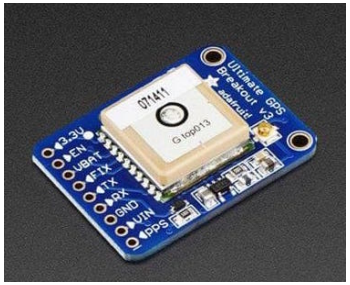
We believe that giving developers insight into **how hardware and software interact** in a richly-linked environment will enable more efficient debugging of embedded systems.

Software

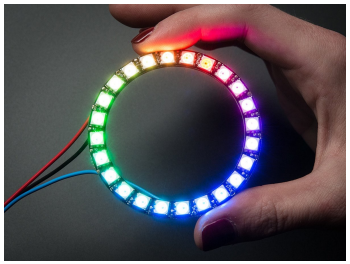




Hardware



Sensors



Actuators

Bifröst

bifröst

Record 5s | New Check

Line
A1
A0 (A)
860
potVal
291

0.1196 0.1197 0.1198 0.1199 0.1200 0.1201 0.1202 0.1203 0.1204 0.1205

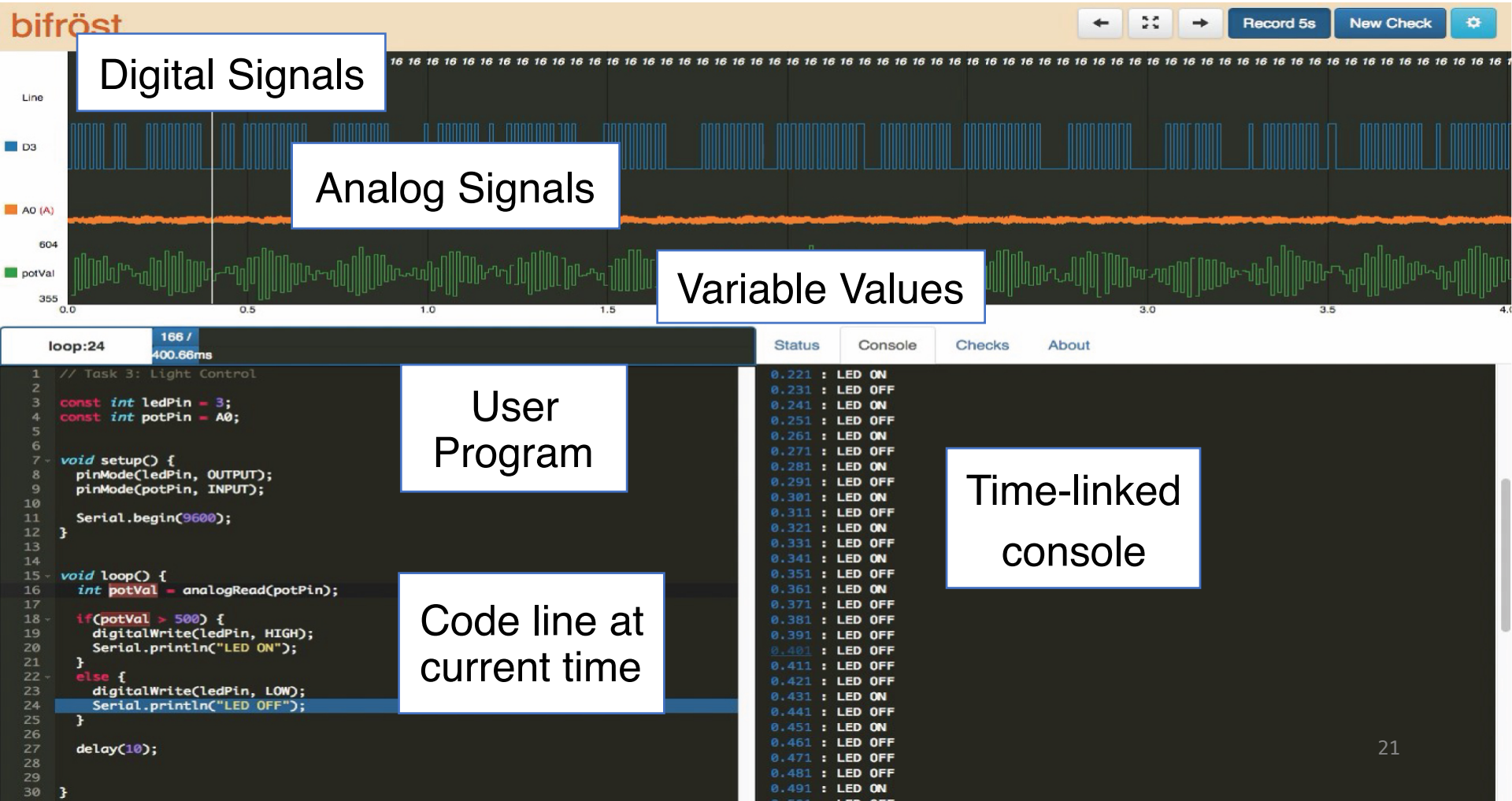
loop:33 | 75 | 120.02ms

```
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(potPin, INPUT);

  // For debug only:
  Serial.begin(9600);
}


void loop() {
  counter++;
  counter = counter % 15;
  int potVal = analogRead(potPin);

  if(potVal > 500) {
    digitalWrite(ledPin, HIGH);

  }
  else {
    digitalWrite(ledPin, LOW);
  }

  delay(10);
  if(counter > 16){
      Serial.print(counter);
  }
  Serial.println(counter);
}
```

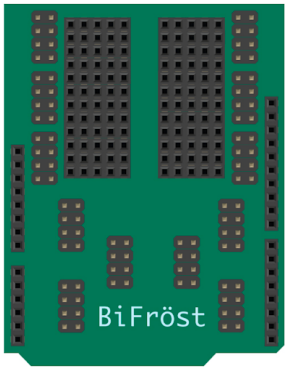
Status | Console | Checks | About

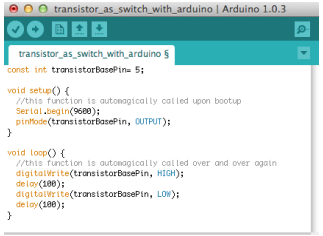
```
[seconds] Beginning serial output...
0.010 : 1
0.020 : 2
0.030 : 3
0.040 : 4
0.050 : 5
0.060 : 6
0.070 : 7
0.080 : 8
0.090 : 9
0.100 : 10
0.110 : 11
0.120 : 12
0.130 : 13
0.140 : 14
0.150 : 0
0.160 : 1
0.170 : 2
0.180 : 3
0.190 : 4
0.200 : 5
0.210 : 6
0.220 : 7
0.230 : 8
0.240 : 9
0.250 : 10
0.260 : 11
0.270 : 12
```

Bifröst: IDE Overview

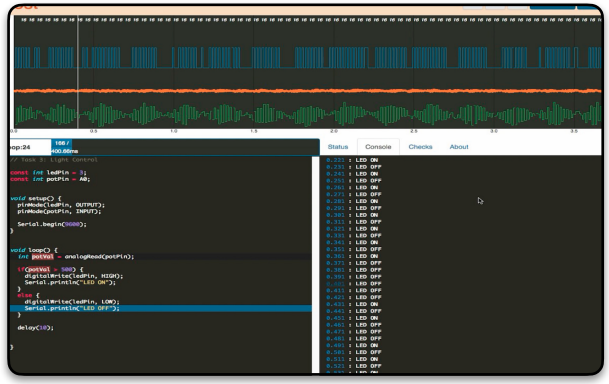


Instrumentation Capture Analysis





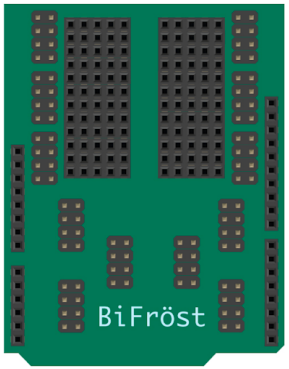
saleae



Instrumentation

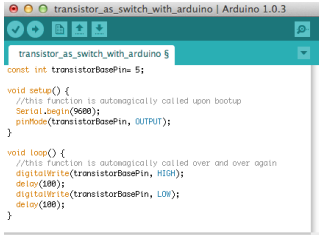
Capture

Analysis



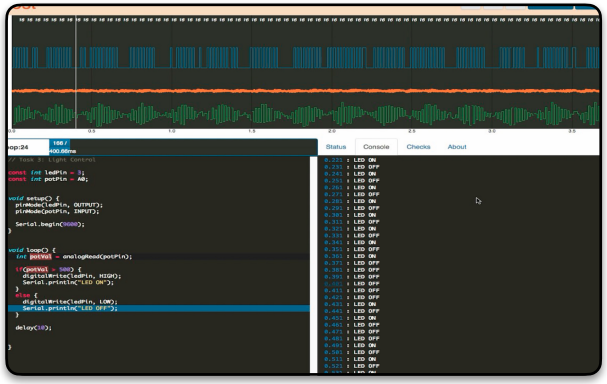
Hardware Instrumentation:

Digital signals
Analog signals



Software Instrumentation:

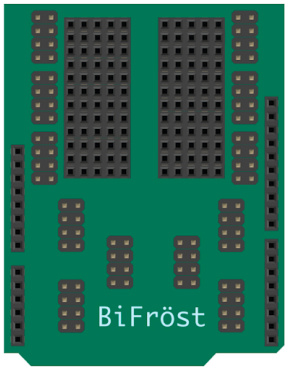
Line numbers
Variable values



Instrumentation

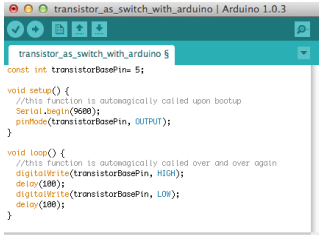
Capture

Analysis



Hardware Instrumentation:

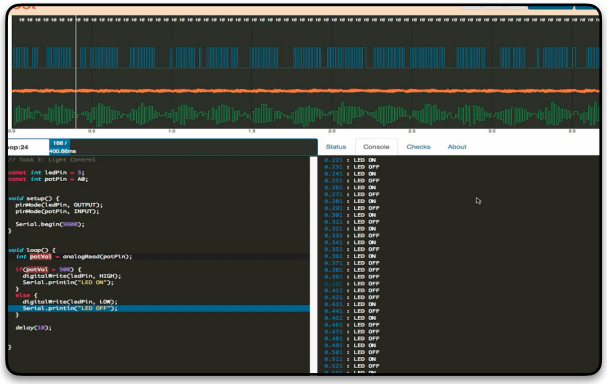
Digital signals
Analog signals



Software Instrumentation:

Line numbers
Variable values

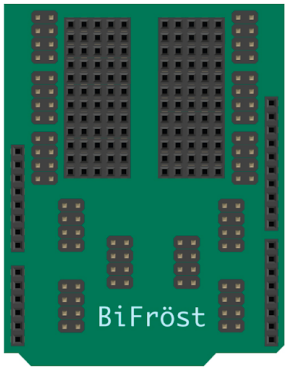
Record with Logic Analyzer



Instrumentation

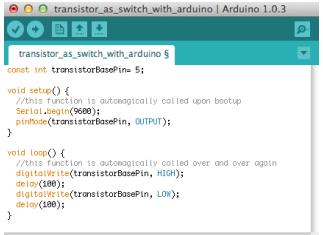
Capture

Analysis



Hardware Instrumentation:

Digital signals
Analog signals

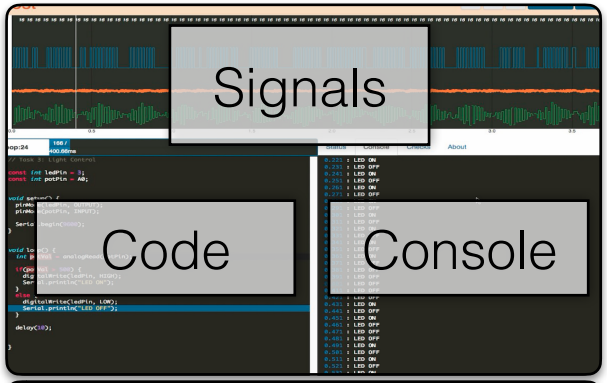


Software Instrumentation:

Line numbers
Variable values

Record with Logic Analyzer

Linked Visualization



Automatic & User Authored Checks

Bifröst: **usage example**

bifröst

Record 5s | New Check

Line

D3

0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

setup:9 | 1 | 0.03ms

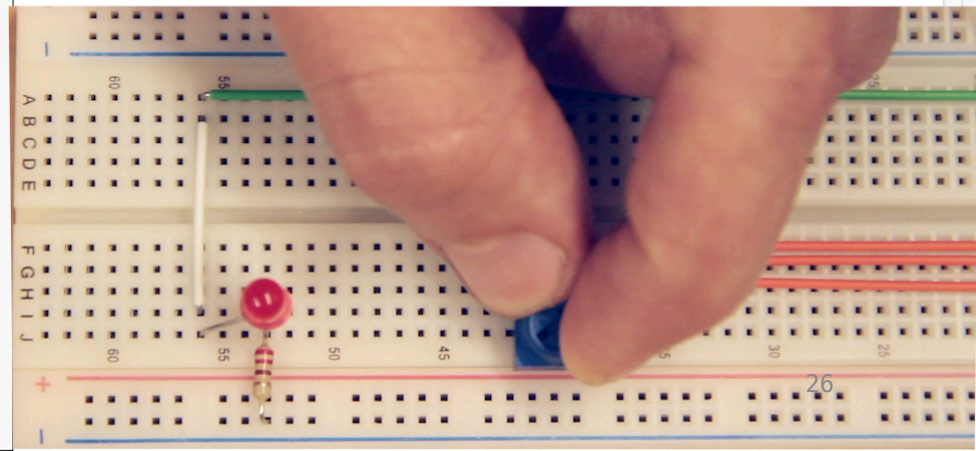
```
// Task 3: Light Control

const int ledPin = A1;
const int potPin = A0;
int counter = 0;

void setup() {

  pinMode(potPin, INPUT);

  // For debug only:
  Serial.begin(9600);
}


void loop() {
  counter++;
  counter = counter % 15;
  int potVal = analogRead(potPin);

  if(potVal > 500) {
    digitalWrite(ledPin, HIGH);

  }
  else {
    digitalWrite(ledPin, LOW);
  }

  delay(10);
  if(counter > 16){
```

Status | Console | Checks | About

+ Errors 1

− Warnings

− Passed



Bifröst: **configuration**

bifröst

Record 5s | New Check

Line

D3

0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

setup:9 | 1 | 0.03ms

```
// Task 3: Light Control

const int ledPin = A1;
const int potPin = A0;
int counter = 0;

void setup() {

  pinMode(potPin, INPUT);

  // For debug only:
  Serial.begin(9600);
}


void loop() {
  counter++;
  counter = counter % 15;
  int potVal = analogRead(potPin);

  if(potVal > 500) {
    digitalWrite(ledPin, HIGH);

  }
  else {
    digitalWrite(ledPin, LOW);
  }

  delay(10);
  if(counter > 16){
```

Status | Console | Checks | About

+ Errors 1

− Warnings

− Passed

Bifröst: **blinking LED**

bifröst

Record 5s | New Check

Line

D3

0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

setup:9 | 1 | 0.03ms

```
// Task 3: Light Control

const int ledPin = A1;
const int potPin = A0;
int counter = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(potPin, INPUT);

  // For debug only:
  Serial.begin(9600);
}


void loop() {
  counter++;
  counter = counter % 15;
  int potVal = analogRead(potPin);

  if(potVal > 500) {
    digitalWrite(ledPin, HIGH);

  }
  else {
    digitalWrite(ledPin, LOW);
  }

  delay(10);
  if(counter > 16){
```

Status | Console | Checks | About

Errors 1

ledPin/D15 – Pin Mode Initialization Before Use

Ensure you set the correct pinMode before reading or writing to any pins.

Available pinMode values are: INPUT, OUTPUT, and INPUT_PULLUP. 528

Warnings

Passed

Bifröst: **recording data**

bifröst

Record 5s | New Check

Line

D3

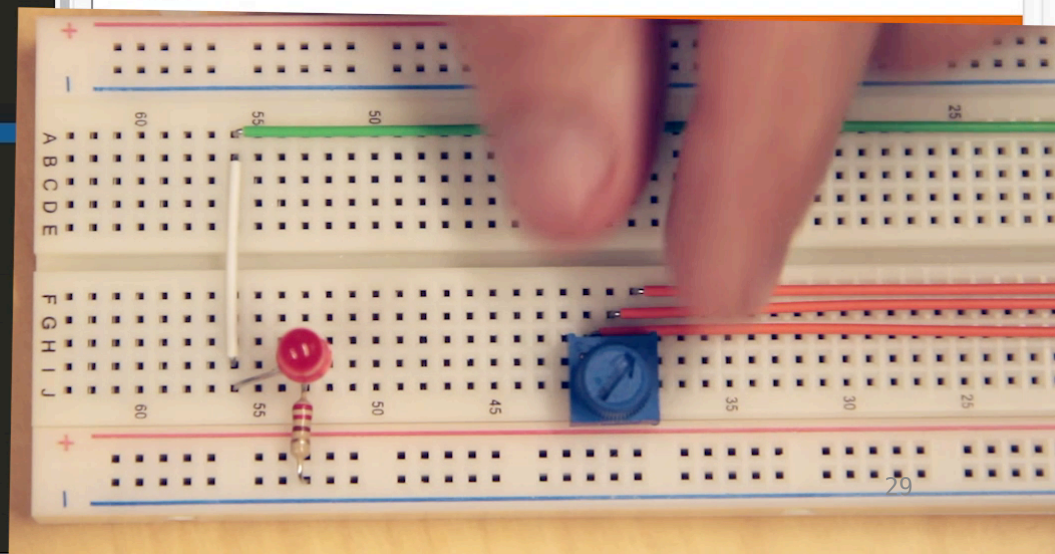
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

setup:9 | 1 | 0.03ms

Status | Console | Checks | About

Errors 1

```
// Task 3: Light Control

const int ledPin = A1;
const int potPin = A0;
int counter = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(potPin, INPUT);

  // For debug only:
  Serial.begin(9600);
}


void loop() {
  counter++;
  counter = counter % 15;
  int potVal = analogRead(potPin);

  if(potVal > 500) {
    digitalWrite(ledPin, HIGH);

  }
  else {
    digitalWrite(ledPin, LOW);
  }

  delay(10);
  if(counter > 16){
```



Bifröst: **variable tracing**

bifröst

Record 5s | New Check

Line

A1

A0 (A)

A1: 1
t: 3.27s

0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

setup:8

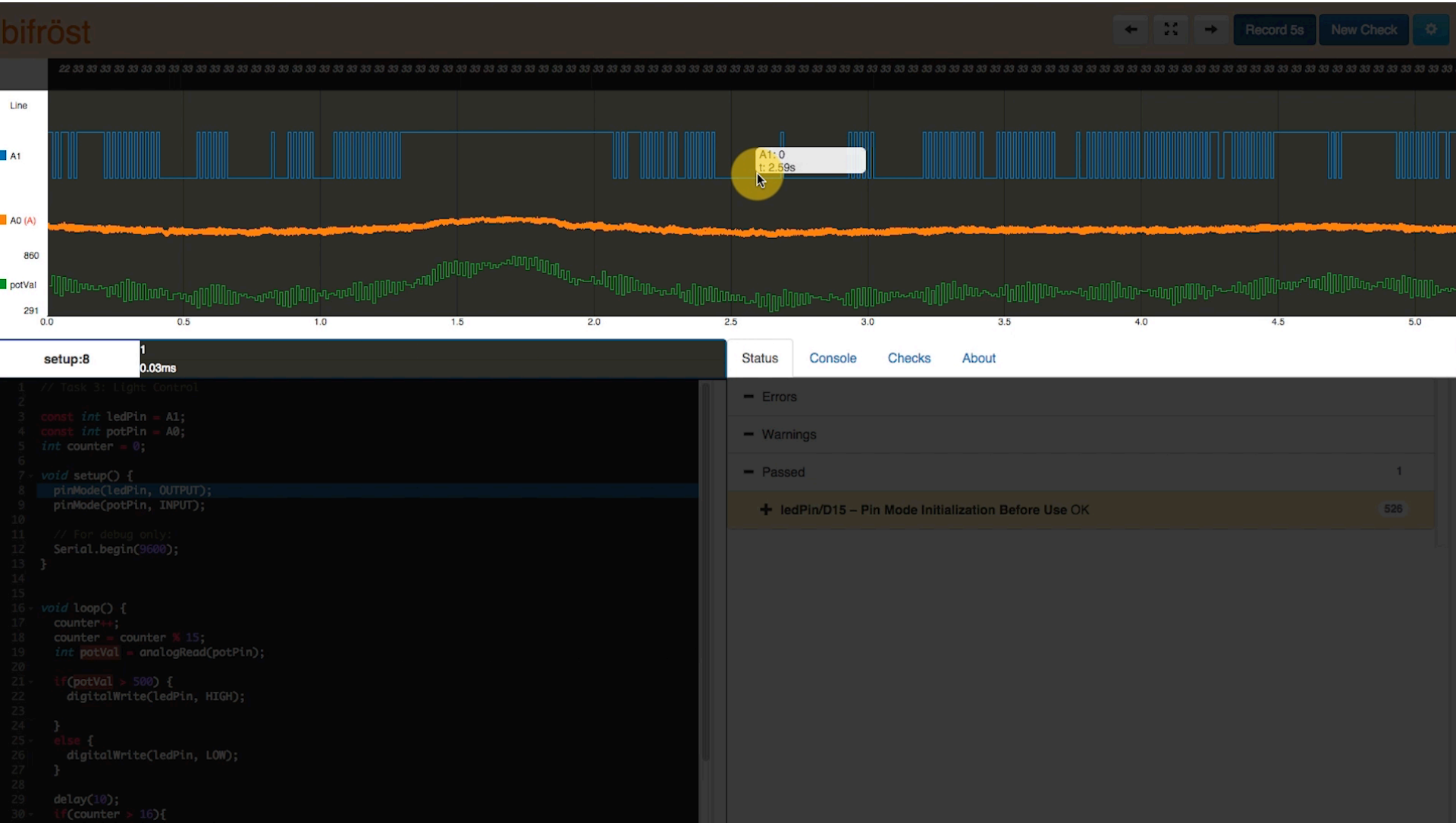
1
0.03ms

```
// Task 3: Light Control

const int ledPin = A1;
const int potPin = A0;
int counter = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(potPin, INPUT);

  // For debug only:
  Serial.begin(9600);
}


void loop() {
  counter++;
  counter = counter % 15;
  int potVal = analogRead(potPin);

  if(potVal > 500) {
    digitalWrite(ledPin, HIGH);

  }
  else {
    digitalWrite(ledPin, LOW);
  }

  delay(10);
  if(counter > 16){
```

Status | Console | Checks | About

- Errors
- Warnings
- Passed 1
  - ledPin/D15 – Pin Mode Initialization Before Use OK 526

Bifröst: **trace ⇔ code linkage**



Bifröst: **code-based navigation**

bifröst

Record 5s | New Check

Line | A1 | A0 (A) | 860 | potVal | 291

19 | 26 | 33

2.592 | 2.593 | 2.594 | 2.595 | 2.596 | 2.597 | 2.598 | 2.599 | 2.600 | 2.601

loop:29 | 1562 / 3158 | 2591.88ms

```
6
7  void setup() {
8    pinMode(ledPin, OUTPUT);
9    pinMode(potPin, INPUT);
10
11   // For debug only:
12   Serial.begin(9600);
13 }
14
15
16 void loop() {
17   counter++;
18   counter = counter % 15;
19   int potVal = analogRead(potPin);
20
21   if(potVal > 500) {
22     digitalWrite(ledPin, HIGH);
23
24   }
25   else {
26     digitalWrite(ledPin, LOW);
27   }
28
29   delay(10);
30   if(counter > 16){
31       Serial.print(counter);
32   }
33   Serial.println(counter);
34 }
35
```

Status | Console | Checks | About

- Errors
- Warnings
- Passed 1
  - + ledPin/D15 – Pin Mode Initialization Before Use OK 526

Bifröst: **console-based navigation**

bifröst

Record 5s | New Check

Line
A1
A0 (A)
860
potVal
291

2.9516 2.9517 2.9518 2.9519 2.9520 2.9521 2.9522 2.9523 2.9524 2.9525

loop:22 | 1777 / 3158 | 2952.06ms

Status | Console | Checks | About

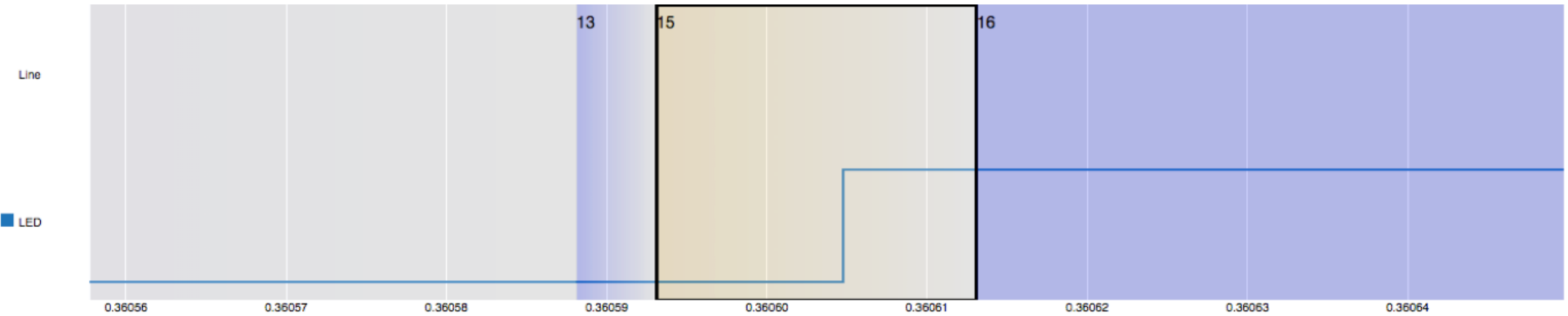
```
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(potPin, INPUT);

  // For debug only:
  Serial.begin(9600);
}


void loop() {
  counter++;
  counter = counter % 15;
  int potVal = analogRead(potPin);

  if(potVal > 500) {
    digitalWrite(ledPin, HIGH);

  }
  else {
    digitalWrite(ledPin, LOW);
  }

  delay(10);
  if(counter > 16){
      Serial.print(counter);
  }
  Serial.println(counter);
}
```

```
[seconds] Beginning serial output...
0.010 : 1
0.020 : 2
0.030 : 3
0.040 : 4
0.050 : 5
0.060 : 6
0.070 : 7
0.080 : 8
0.090 : 9
0.100 : 10
0.110 : 11
0.120 : 12
0.130 : 13
0.140 : 14
0.150 : 0
0.160 : 1
0.170 : 2
0.180 : 3
0.190 : 4
0.200 : 5
0.210 : 6
0.220 : 7
0.230 : 8
0.240 : 9
0.250 : 10
0.260 : 11
```

Data Capture

bifröst

[loop:15] [10/19

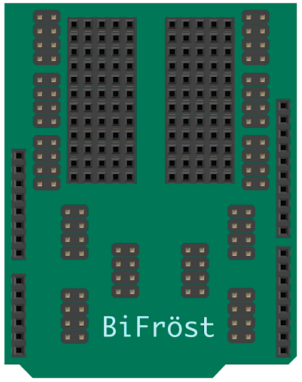


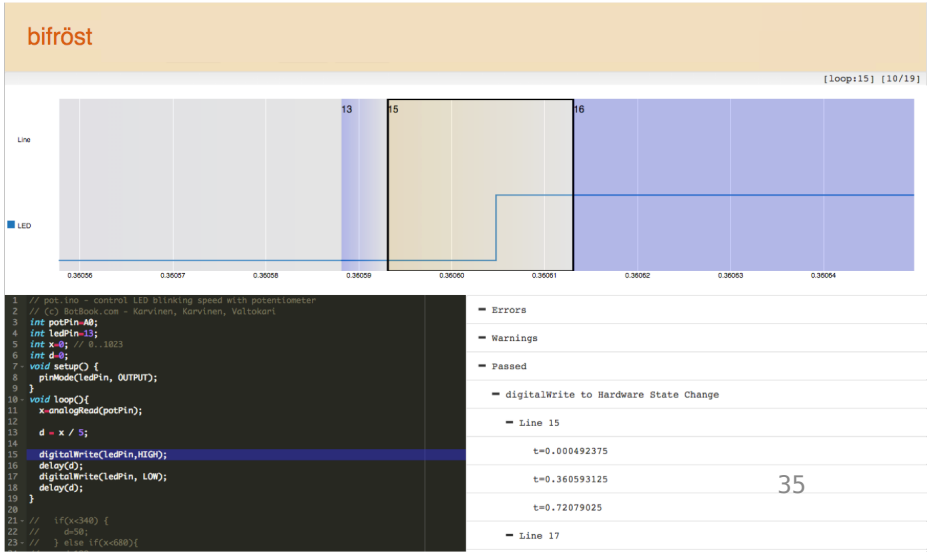

```
1  // pot.ino - control LED blinking speed with potentiometer
2  // (c) BotBook.com - Karvinen, Karvinen, Valtokari
3  int potPin=A0;
4  int ledPin=13;
5  int x=0; // 0..1023
6  int d=0;
7  void setup() {
8    pinMode(ledPin, OUTPUT);
9  }
10 void loop(){
11   x=analogRead(potPin);
12
13   d = x / 5;
14
15   digitalWrite(ledPin,HIGH);
16   delay(d);
17   digitalWrite(ledPin, LOW);
18   delay(d);
19 }
20
21 //   if(x<340) {
22 //     d=50;
23 //   } else if(x<680){
```

- Errors
- Warnings
- Passed
  - digitalWrite to Hardware State Change
    - Line 15
      - t=0.000492375
      - t=0.360593125
      - t=0.72079025
    - Line 17

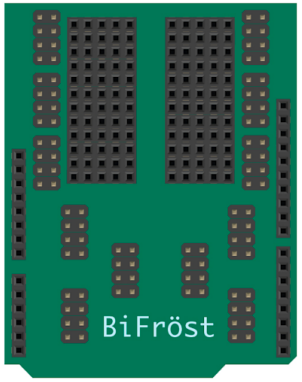
Capture: **example**

Backend

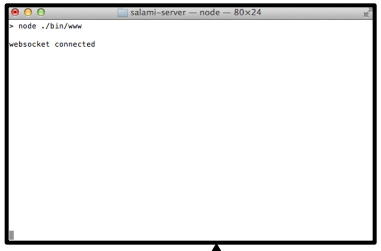




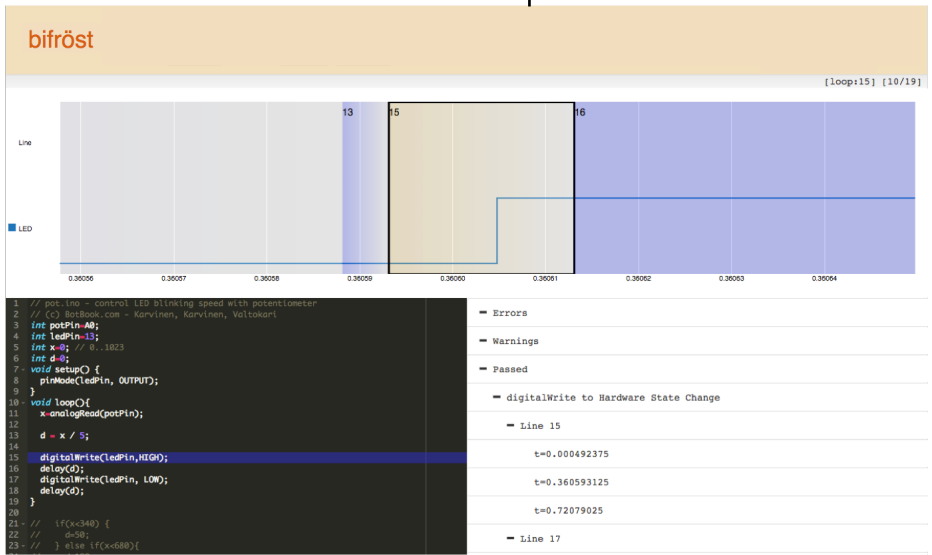
Capture: **setup**



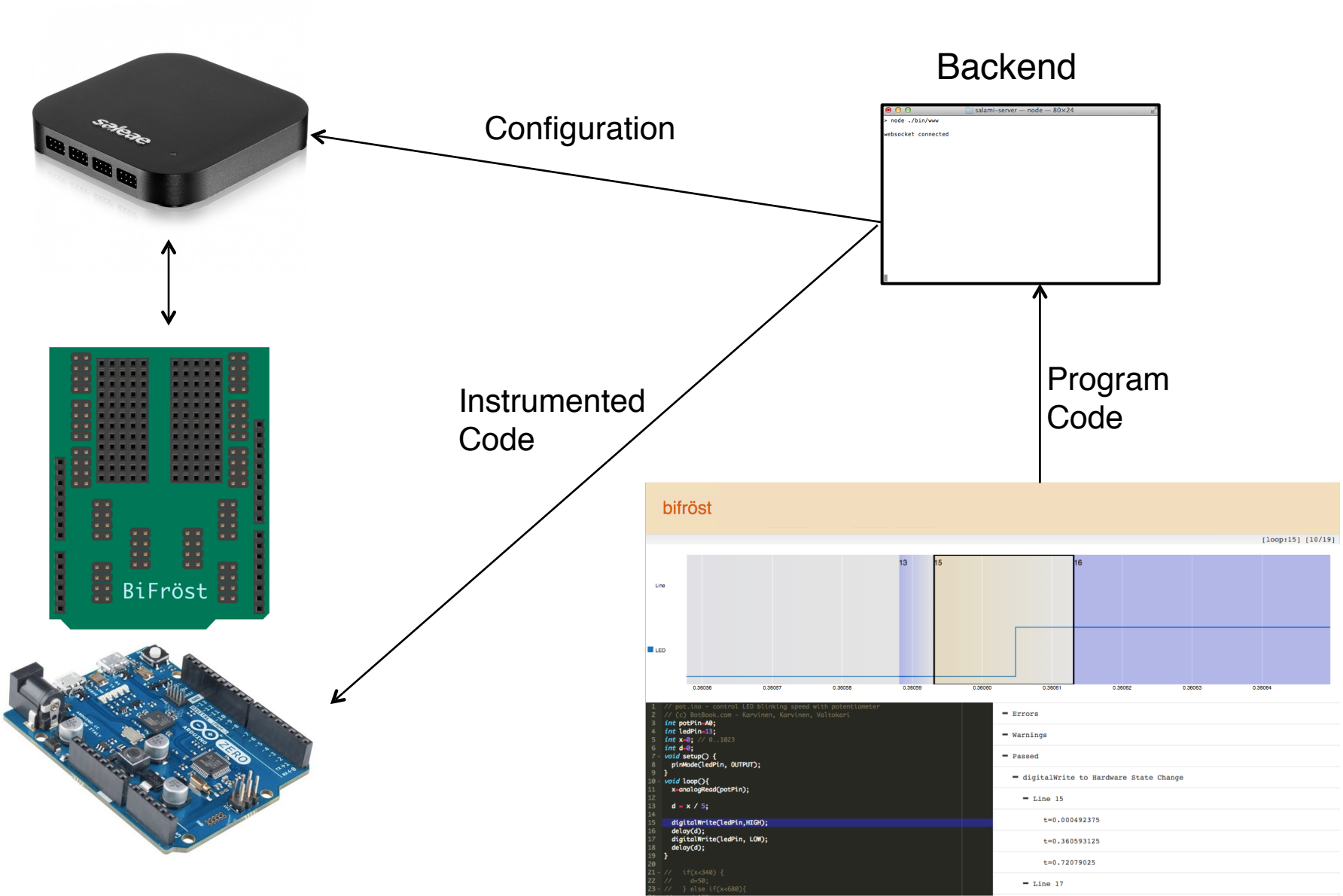
Backend



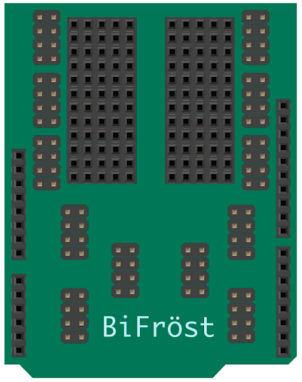
Program Code



Capture: **setup**

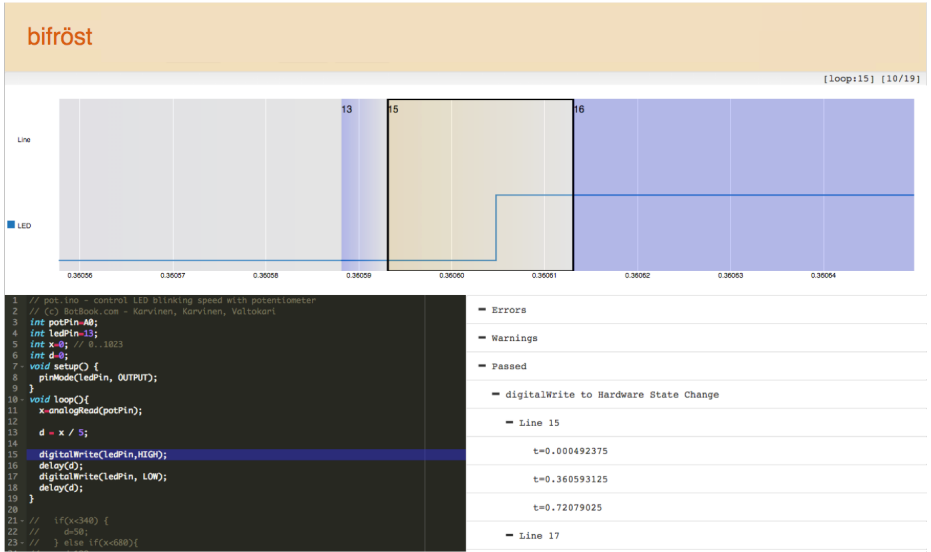


Capture: **execution**



Debugging information: *code lines + function info*

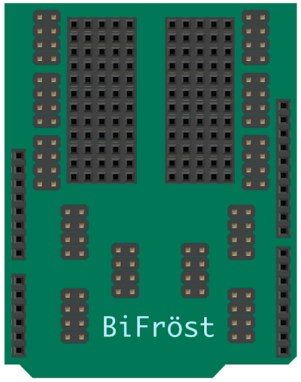
Backend



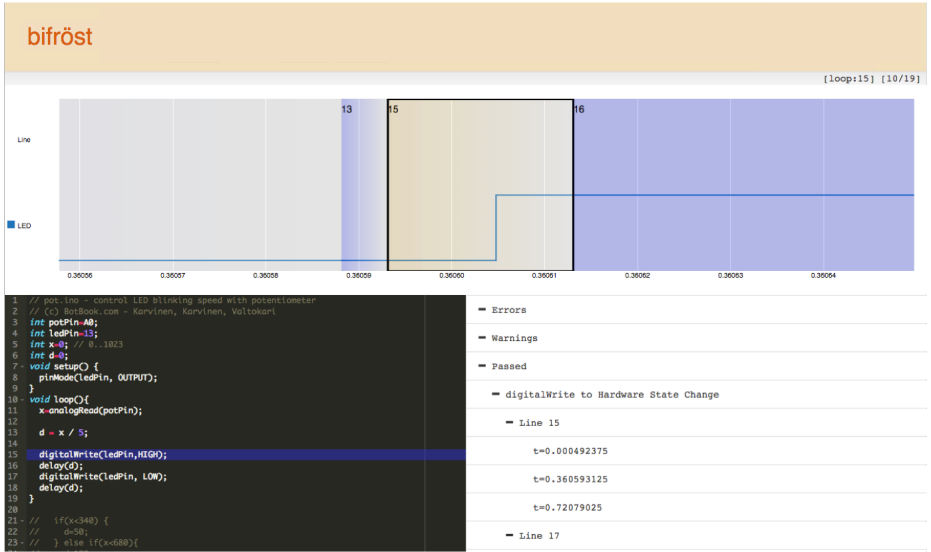
Capture: **execution**

Trace (w/ debug info) + decoded protocols

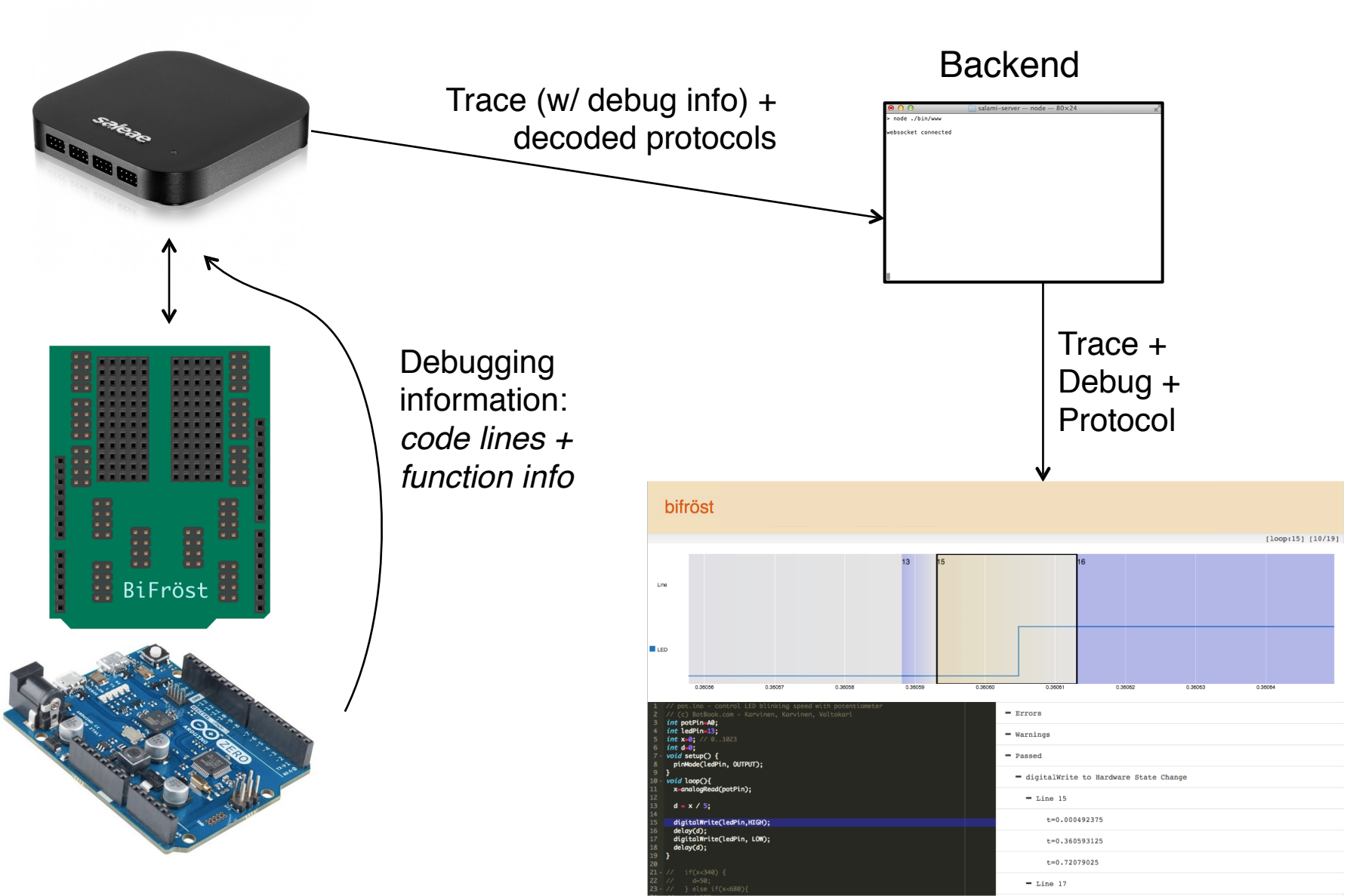
Backend



Debugging information: *code lines + function info*



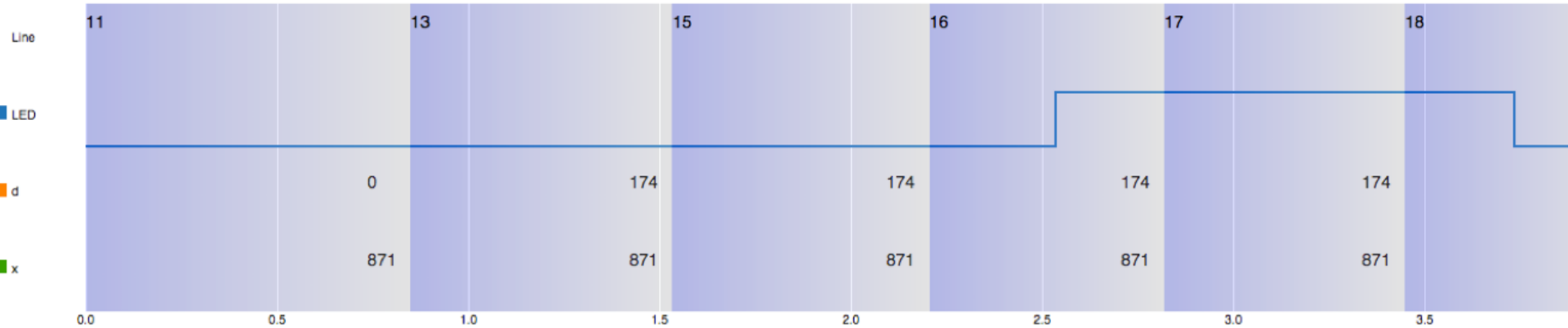
Capture: **execution**



Capture: **inspection**

incrementally *step*, *record*, and *visualize*

bifröst




```
// pot.ino - control LED blinking speed with potentiometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int potPin=A0;
int ledPin=13;
int x=0; // 0..1023
int d=0;
void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop(){
```

- Errors
  - + digitalWrite to Hardware State Change
- Warnings
  - + digitalWrite to Hardware State Change

Code Instrumentation

```
43 void readRegister(char registerAddress, int numBytes, signed char * values){
44   //Since we're performing a read operation, the most significant bit of the register address should be set.
45   char address = 0x80 | registerAddress;
46   //If we're doing a multi-byte read, bit 6 needs to be set as well.
47   if(numBytes > 1)address = address | 0x40;
48
49   //Set the Chip select pin low to start an SPI packet.
50   digitalWrite(CS, LOW);
51   //Transfer the starting register address that needs to be read.
52   SPI.transfer(address);
53   //Continue to read registers until we've read the number specified, storing the results to the input buffer.
54   for(int i=0; i<numBytes; i++){
55     values[i] = SPI.transfer(0x00);
56   }
57   //Set the Chips Select pin high to end the SPI packet.
58   digitalWrite(CS, HIGH);
59 }
```

Code Instrumentation

```
47 void readRegister(char registerAddress, int numBytes, signed char * values){
48   //Since we're performing a read operation, the most significant bit of the register address should be set.
49   Serial.write(45);
50   char address = 0x80 | registerAddress;
51   //If we're doing a multi-byte read, bit 6 needs to be set as well.
52   Serial.write(47);
53   if(numBytes > 1)address = address | 0x40;
54
55   //Set the Chip select pin low to start an SPI packet.
56   Serial.write(50);
57   digitalWrite(CS, LOW);
58   //Transfer the starting register address that needs to be read.
59   Serial.write(52);
60   SPI.transfer(address);
61   //Continue to read registers until we've read the number specified, storing the results to the input buffer.
62   for(int i=0; i<numBytes; i++){
63     Serial.write(55);
64     values[i] = SPI.transfer(0x00);
65   }
66   //Set the Chips Select pin high to end the SPI packet.
67   Serial.write(58);
68   digitalWrite(CS, HIGH);
69 }
```

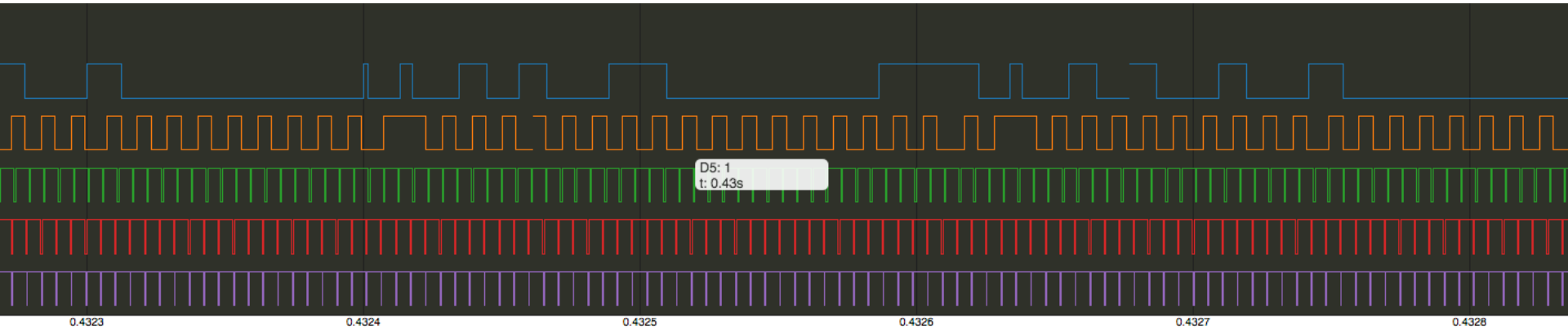
Core Library Instrumentation

```
1  void digitalWrite( uint32_t ulPin, uint32_t ulVal )
2  {
3
4    Serial1.write(193);  // Opcode for digitalWrite
5    Serial1.write(ulPin); // Pin
6    Serial1.write(ulVal); // Output Value
7    // Handle the case the pin isn't usable as PIO
8    if ( g_APinDescription[ulPin].ulPinType == PIO_NOT_A_PIN )
9    {
10     return ;
11   }
12
13   // Enable pull-up resistor
14   PORT->Group[g_APinDescription[ulPin].ulPort].PINCFG[g_APinDescript
15
```

Checks

automatically inferred

user-parameterized



Checks: **automatically inferred**

"Writes should generate activity on a pin"

"Communication protocols adherence"

Status | Console | Checks | About

+ blinkyLED/D4 – Software Write to Hardware State Change
Your microcontroller wrote a value to a pin, but the pin did not change to that value. You should check whether the pin has the correct pinMode setup in the `setup()` function, and that there is no other connection to the pin which would stop it from changing. 10

− Warnings 1

+ buttonLED/D5 – Software Write to Hardware State Change
Pin was already set to the specificied value when written to. 31

− Passed 3

+ buttonLED/D5 – Pin Mode Initialization Before Use OK 53

Checks: **user-parameterized**

"when D1 rises expect line 11 in 100 ms"

Checks: **user-parameterized**

"when D1 rises expect line 11 in 100 ms"

pin

Checks: **user-parameterized**

"when D1 rises expect line 11 in 100 ms"

pin *event*

Checks: **user-parameterized**

"when D1 rises expect line 11 in 100 ms"

pin *event* *source code*

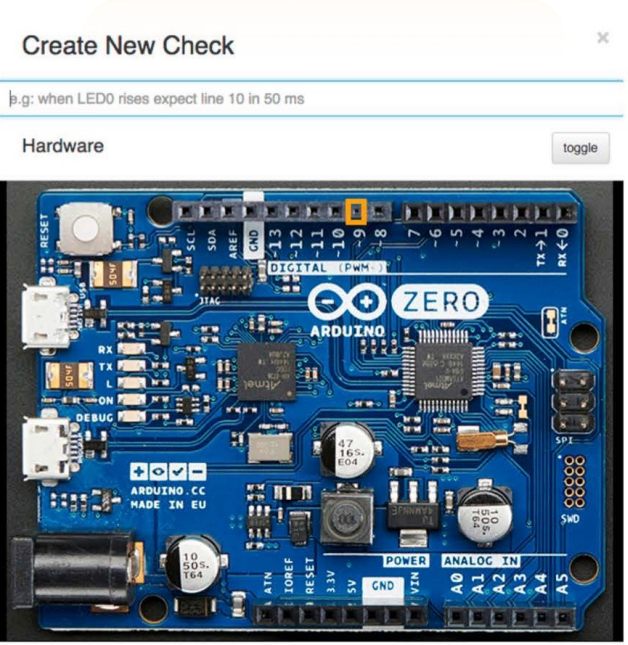
Checks: **user-parameterized**

"when D1 rises expect line 11 in 100 ms"

pin *event* *source code* *timing*

Checks: **user-parameterized**

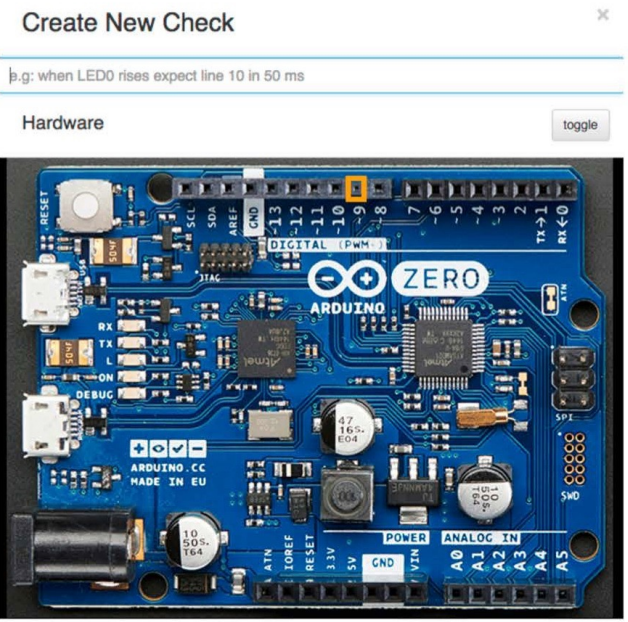
Hardware → Trigger → Software

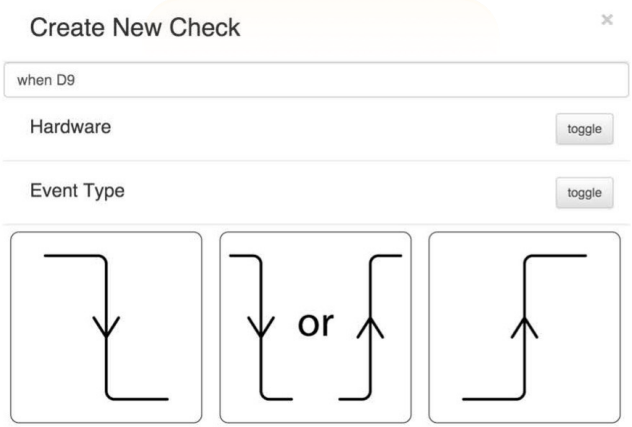


"When D1 rises, expect Line 17 in 20 ms"

Checks: **user-parameterized**

Hardware → Trigger → Software

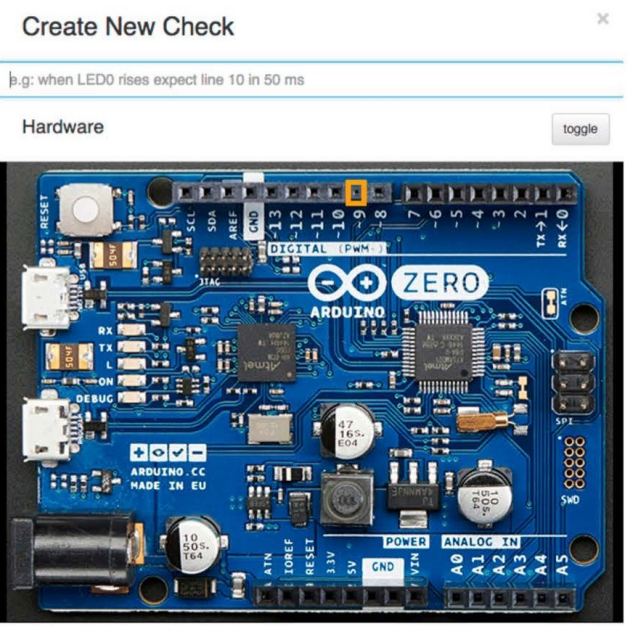


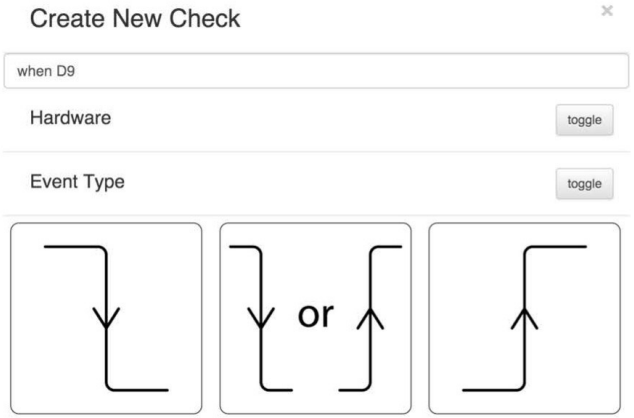


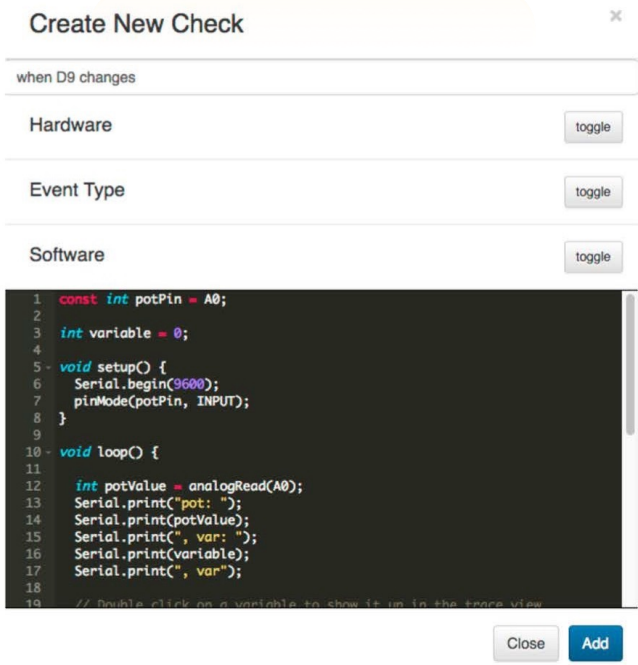
"When D1 rises, expect Line 17 in 20 ms"

Checks: **user-parameterized**

Hardware → Trigger → Software



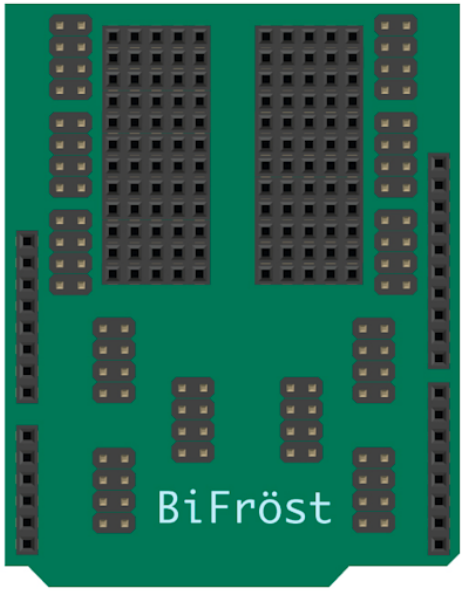




"When D1 rises, expect Line 17 in 20 ms"

Evaluation

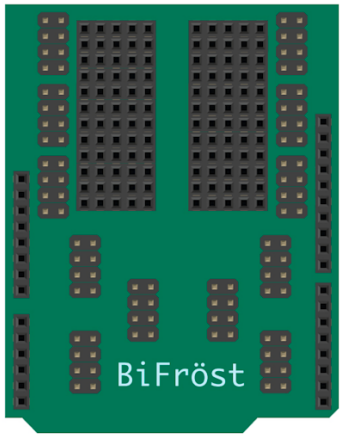
How effectively does Bifröst aid embedded debugging?



Evaluation: **issues**

Users identified issues such as:

pin misconfiguration
thresholding issues
switch bouncing
timing issues



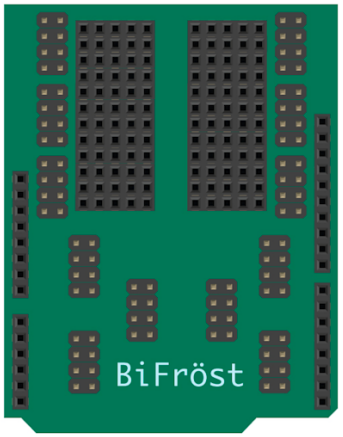
Evaluation: **strategies**

Users identified these issues using:

variable/signal data comparison

code statement execution monitoring

automatic and user-inferred checks



Limitations

performance:

instrumentation overhead
(full speed to under 10x)

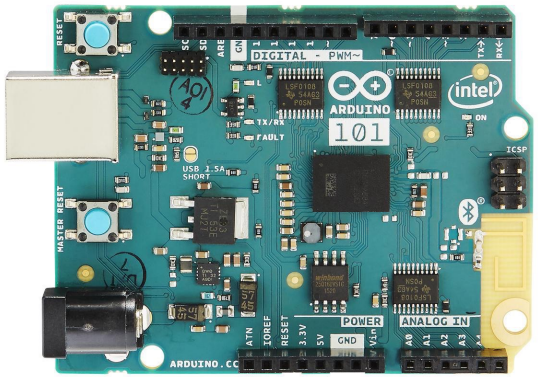
scope:

only instruments user code
and Arduino core functions

```
void readRegister(char registerAddress,
  //Since we're performing a read operat
  Serial.write(45);
  char address = 0x80 | registerAddress;
  //If we're doing a multi-byte read, bi
  Serial.write(47);
  if(numBytes > 1)address = address | 0x

  //Set the Chip select pin low to start
  Serial.write(50);
  digitalWrite(CS, LOW);
  //Transfer the starting register addre
  Serial.write(52);
  SPI.transfer(address);
  //Continue to read registers until we'
  for(int i=0; i<numBytes; i++){
    Serial.write(55);
    values[i] = SPI.transfer(0x00);
  }
  //Set the Chips Select pin high to end
  Serial.write(58);
  digitalWrite(CS, HIGH);
}
```

Future Work

- Support for continuous capture
- Support for multiple devices
- Checking & visualizing networked behavior



Thanks!

Will McGrath – wmcgrath@stanford.edu
Jeremy Warner – jeremy.warner@berkeley.edu

With: Daniel Drew, Mitchell Karchemsky, Majeed Kazemitabaar, David Mellis, and Björn Hartmann