Embedded Systems Reverse Engineering

// WEEK 09

Operators in Embedded Systems:
Debugging and Hacking Operators
w/ DHT11 Sensor Single-Wire Protocol

George Mason University

RP2350 // ARM Cortex-M33

C Operators Overview

Six Types of Operators in C

Arithmetic

`+ - * / %`

Math operations

5 * 10 = 50

Increment

`x++ ++x x--`

Add/subtract by 1

x++ returns old val

Relational

`> < >= <= == !=`

Compare values

(6 > 10) = false

Logical

`&& || !`

Combine conditions

AND, OR, NOT

Bitwise

`<< >> & | ^ ~`

Manipulate bits

6 << 1 = 12

Assignment

`+= -= *= /=`

Assign and modify

x += 5 (x=x+5)

This Week's Program

0x001a_operators.c demonstrates all 6 types

DHT11 temperature/humidity sensor + operator calculations

KEY: Compiler pre-computes constant expressions

In the binary, most operators become immediate values

Arithmetic & Increment

Math Operations and Post/Pre Increment

Arithmetic Operators

+	5 + 10 = 15	Addition
-	10 - 5 = 5	Subtraction
*	5 * 10 = 50	Multiplication
/	10 / 5 = 2	Division
%	10 % 3 = 1	Modulus

Post vs Pre Increment

Post: x++

Use value THEN increment

a = x++ --> a=5, x=6

Pre: ++x

Increment THEN use value

b = ++x --> x=7, b=7

Post-Increment Step by Step

```
int x = 5;
int result = x++;
```

Step 1: result = x — **result gets 5**

Step 2: x = x + 1 — **x becomes 6**

Final: result = 5   x = 6   "Use first, THEN increment"

In our code: `int increment_operator = x++;`

x was 5, so increment_operator = 5, then x becomes 6

Relational & Logical

Comparing Values and Combining Conditions

Relational Operators

Compare two values --> true (1) or false (0)

>	6 > 10	false	Greater than
<	6 < 10	true	Less than
>=	6 >= 6	true	Greater/equal
<=	6 <= 10	true	Less or equal
==	6 == 10	false	Equal to
!=	6 != 10	true	Not equal

Logical Operators

Combine conditions into one result

&&	AND -- both must be true
||	OR -- at least one true
!	NOT -- inverts result

AND Truth Table

A	B	A && B
false	false	false
false	true	false
true	false	false

In Our Code (x=6, y=10)

```
bool relational = (x > y);          (6 > 10) = false = 0
bool logical = (x>y) && (y>x);      false && true = false = 0
```

In the binary: Both compile to immediate #0

Compiler pre-computes: constants are known at compile time
Result 0 = false, Result 1 = true

Bitwise & Assignment

Bit Manipulation and Compound Assignment

Bitwise Operators

<<	6 << 1 = 12	Left shift
>>	6 >> 1 = 3	Right shift
&	6 & 3 = 2	AND
|	6 | 3 = 7	OR
^	6 ^ 3 = 5	XOR
~	~6	NOT (invert)

Left shift = multiply by 2

```
0 0 0 0 0 1 1 0     = 6
0 0 0 0 1 1 0 0     = 12
```

Assignment Operators

Shorthand for math + assign

+=	x += 5	x = x + 5
-=	x -= 2	x = x - 2
*=	x *= 3	x = x * 3
/=	x /= 2	x = x / 2
%=	x %= 4	x = x % 4

In our code (x=6 after x++):

x += 5 --> 6 + 5 = 11

In Our Code (x=6, y=10)

```
int bitwise = (x<<1);          6 << 1 = 12 (0b0110 --> 0b1100)
```

Expected Output

bitwise_operator: 12   assignment_operator: 11

Both pre-computed by compiler as immediates

DHT11 Sensor

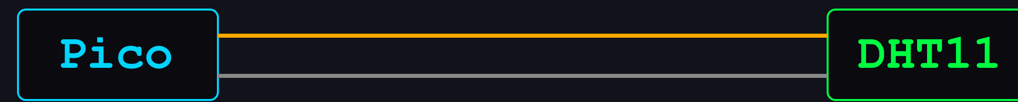
Single-Wire Temperature and Humidity

DHT11 Pinout

1:VCC 2:DATA 3:NC 4:GND

Humidity: 20-90% RH (+/-5%)
Temp: 0-50C (+/-2C)
Protocol: custom one-wire

Wiring to Pico 2



GPIO 4 = DATA
3.3V = VCC
GND = GND

1. Host pulls LOW 18ms
2. DHT11 responds, sends 40 bits

Source Code: 0x001a_operators.c

```c
int x = 5, y = 10;
int arithmetic = (x * y);          // 50
int increment = x++;               // 5 (post)
bool relational = (x > y);         // false
bool logical = (x>y)&&(y>x);       // false
int bitwise = (x<<1);              // 12
int assignment = (x += 5);         // 11
float hum, temp;
dht11_read(&hum, &temp);
```

Variable Flow

Tracing x Through Every Operator

Tracing x Step-by-Step

Line	x	Result
`int x = 5, y = 10;`	5	x initialized to 5
`int arithmetic = (x * y);`	5	arithmetic = 50
`int increment = x++;`	5-->6	increment = 5 use THEN increment
`bool relational = (x > y);`	6	relational = false 6 > 10 is false
`bool logical = (x>y)&&(y>x);`	6	logical = false false AND true = false
`int bitwise = (x<<1);`	6	bitwise = 12 0b0110 << 1 = 0b1100
`int assignment = (x += 5);`	6-->11	assignment = 11 6 + 5 = 11

DHT11 Output

Humidity: 51.0% **Temperature: 23.8C**

dht11_read(&hum, &temp) -- passes addresses so function can write values

Vector Table

Finding Reset_Handler and main()

ARM Vector Table

Base address: 0x10000000

Offset	Contents	Purpose
0x00	Initial SP	Stack ptr
0x04	Reset_Handler	Entry point
0x08	NMI_Handler	NMI
0x0C	HardFault	Fault

Decoding the Address

At 0x10000004:

Bytes: 5d 01 00 10

Step 1: Reverse (little-endian)

10 00 01 5d = 0x1000015d

Step 2: Remove Thumb bit

0x1000015d - 1 = 0x1000015c

Reset_Handler --> main()

Reset_Handler at 0x1000015c calls 3 functions:

Call 1: some_init()	Hardware initialization
Call 2: main()	**THIS IS WHAT WE WANT** Address: 0x10000234
Call 3: exit()	Never returns

The MIDDLE function call is always main()

Navigate to 0x10000234 in Ghidra to find it

IEEE-754 Floats

How Computers Store Decimal Numbers

32-bit Float Structure

S	Exponent	Mantissa (Fraction)
1 bit	8 bits	23 bits

Value = (-1)^S x (1 + Mantissa) x 2^(Exponent - 127)

Example: Decoding 0.1f

Little-endian bytes: **cd cc cc 3d**

Reversed (big-endian): **0x3dcccccd**

Sign: 0 **Exp: 01111011 = 123** **Mantissa: 1001100...**

Exp - 127 = -4, so value = 1.6 x 2^(-4) **= 0.1**

IEEE-754 Quick Reference

Value	Hex	Bytes (LE)
0.1	0x3dcccccd	cd cc cc 3d
1.0	0x3f800000	00 00 80 3f
5.0	0x40a00000	00 00 a0 40
10.0	0x41200000	00 00 20 41
-1.0	0xbf800000	00 00 80 bf

Hacking the Float

Changing the DHT11 Scaling Constant

DHT11 Scaling Calculation

result = integer + (decimal x 0.1)

0.1f is our target!

Example: temp = 23 + (8 x 0.1) = 23.8C

Key Offsets in Binary

Offset	Bytes	Meaning
0x410	a6 ee 25 7a	vfma.f32 s14,s12,s11 (humidity)
0x414	e6 ee a5 7a	vfma.f32 s15,s13,s11 (temp)
0x42C	cd cc cc 3d	0.1f -- the scaling constant

The Hack: 0.1f --> 5.0f

At offset 0x42C, change:

Original: cd cc cc 3d (0.1f)

Patched: 00 00 a0 40 (5.0f)

New result: 23 + (8 x 5.0) = 63.0C

Decimal part is now multiplied by 5.0 instead of 0.1

Export .bin from Ghidra, convert to UF2, flash to Pico

Operators & DHT11 Hacking

Operators, DHT11, IEEE-754, and Hacking

6 Operator Types

Arithmetic	x * y = 50
Increment	x++ returns 5, x becomes 6
Relational	(6 > 10) = false
Logical	false && true = false
Bitwise	6 << 1 = 12
Assignment	x += 5 = 11

Post-increment: use THEN increment

Key Addresses

0x10000000	Vector table
0x10000004	Reset_Handler addr
0x10000234	main()
0x10000410	Humidity vfma
0x10000414	Temp vfma
0x1000042C	0.1f constant (hack)

IEEE-754 Format

S(1) + Exp(8) + Mantissa(23)

(-1)^S x (1+M) x 2^(E-127)

0.1f = 0x3dcccccd = cd cc cc 3d

Hack Workflow

1. Analyze in Ghidra
2. Find float at 0x42C
3. Patch cd cc cc 3d

Binary Hacking Steps

Analyze --> Identify --> Offset --> Patch --> Export --> Test

Project: 0x001a_operators

Source: 0x001a_operators.c with DHT11 sensor on GPIO 4