Lab 07

Strain Gauge Measurements

Jordan Pinson - 1002209221
Bryan Gonzalez - 1001443032
CSE 4355- Section 001
Dr. Jason Losh
Fall 2025

TM4C C Code

```
//-----------------------------------------------------------------------------
// Subroutines
//-----------------------------------------------------------------------------

void init_gpio()
{
    /*Enable clocks*/
    SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_R3;    //PORT D Clock
    _delay_cycles(3);

    /*Port D config*/
    GPIO_PORTD_DIR_R |= PD_CLOCK_MASK;  //SET P_CLK AS OUTPUT
    GPIO_PORTD_DIR_R &= ~DATA_MASK;     //SET DATA AS INPUT
    GPIO_PORTD_DEN_R |= PD_CLOCK_MASK | DATA_MASK;
}

void initHW()
{
    initSystemClockTo40Mhz();
    initUart0();
    init_gpio();
    init_tLED();
}
```

```
char* toAsciiHex(char *buff, uint32_t Val)
{
    uint32_t hex_digit = 0;
    int i;

    for(i = 0; i < 8; i++)
    {
        //Shifts bits in groups of 4 towards the lsb and isolates them using our 0xF flag
        hex_digit = Val >> (4 * (7-i));
        hex_digit &= 0xF;

        if(hex_digit < 10)  //hex value is 0-9
            buff[i] = 48 + hex_digit;
        else                //hex value is a-b (starts at 97 but we need to shift by 10 starting at a)
            buff[i] = 87 + hex_digit;
    }
    buff[8] = '\0';

    return buff;
}

char* toAsciiDecimal(char *buff, uint32_t Val)
{
    uint32_t digit = 0;
    int i;

    for(i = 14; i >= 0; i--)
    {
        if(Val == 0)
            buff[i] = ' ';
        else
        {
            digit = Val % 10;
            Val /= 10;
            buff[i] = '0' + digit;
        }

    }
    buff[15] = '\0';

    return buff;
```

```
//-------------------------------------------------------------------------------
// Main
//-------------------------------------------------------------------------------

int main(void)
{
    initHW();

    PD_CLK = 0;
    int32_t data = 0;
    char buffer[128];

    while(1)
    {
       data = 0;
       while(DATA);
       int i;
       for(i = NUM_BITS-1; i >= 0; i--)
       {
           PD_CLK = 1;
           waitMicrosecond(WAIT_TIME);
           data |= (DATA << i);

           PD_CLK = 0;
           waitMicrosecond(WAIT_TIME);
       }

       PD_CLK = 1;
      waitMicrosecond(WAIT_TIME);
```

```
        PD_CLK = 0;
        waitMicrosecond(WAIT_TIME);
        putsUart0("\033[2J\033[H");
        putsUart0("Lab 7 - Strain Guarge \n\r");
        putsUart0("Raw data: \t     0x");
        toAsciiHex(buffer, data);
        putsUart0(buffer);
        putsUart0("\n\r\t\t");
        toAsciiDecimal(buffer, data);
        putsUart0(buffer);
        putsUart0("\r\n");

        float Grams = decToGrams(data);
        if(Grams < 0) Grams = 0;
        toAsciiDecimal(buffer, (int)Grams);
        putsUart0("Grams: \t\t");
        putsUart0(buffer);
        putsUart0("g \n\r");

        float Newtons = gramsToNewt(Grams);
        toAsciiDecimal(buffer, (int)Newtons);
        putsUart0("Newtons: \t");
        putsUart0(buffer);
        putsUart0("N \n\r");


    }
        return 0;
}
```

```
float decToGrams(int32_t raw)
{
    return -0.02350849851 * (float)raw + (float)305469;
}

float gramsToNewt(float grams)
{
    float kGrams = grams / 1000;
    return kGrams * 9.81;
}
```

Code Explanation

1. Initializes the system clock, UART0, and Port D GPIO.
    - `GPIO_PORTD_DIR_R |= PD_CLOCK_MASK;` sets PD1 as an output (clock).
    - `GPIO_PORTD_DIR_R &= ~DATA_MASK;` sets PD0 as an input (data).
2. Bit-banded aliases `DATA` and `PD_CLK` so you can read/write the single bit directly.
3. Main Loop:
    - `while(DATA);` waits for the DATA pin to go **low** (the loop executes while DATA is nonzero). This effectively waits until DATA == 0.
        - **HX711** behavior: DOUT goes **LOW** when data is ready (so the code is waiting for the ready-low).
    - Then it clocks 24 bits by toggling PD_CLK and reading the DATA bit, building a 24-bit value into `data`.
    - Pulses one more clock to choose channel/gain (25th clock).
    - Prints the raw `data` to UART using `toAsciiDecimal` (twice).

ADC → Force Conversion

1. Tare/Zero
    - With nothing applied to the beam, take N samples and average them to get raw_zero (removes offset/noise).
2. Apply known masses
    - Hang a set of known masses (e.g., 50 g, 100 g, 200 g) at the same measurement point and record the average ADC reading for each mass: raw_i.
    - For each known mass m_i (grams), compute the corresponding force F_i = m_i / 1000 * 9.80665 (N) if you want force directly.
3. Compute sensitivity
    - For mass calibration (counts per gram):
      Counts_per_gram = (raw_with_mass − raw_zeromass_g) / mass_g

      *Uses a linear least-squares fit across multiple points to get better accuracy
    - For force calibration (counts per Newton):
      counts_per_N = (raw_with_mass − raw_zero) / F_N
4. Conversion equations
    - Mass (grams):
      mass_g = (raw_meas − raw_zero) / counts_per_gram
    - Force (Newton):
      F_N = (raw_meas − raw_zero) / counts_per_N

      *Equivalently F_N = (mass_g / 1000) * g.

Observed Behavior

Behaviour: Pressing the end of the beam trying to “shortening” it, the reading will indicate a force/strain that’s inconsistent with the bending calibration.

Cause: Current single-gauge (or single side of bridge) arrangement measures bending strain at that gauge location. Axial compression/tension of the beam (shortening) produces a different strain component (axial), and depending on where the gauge is placed it can add or subtract from the bending signal. In short: we are mixing bending strain and axial strain and we haven’t instrumented the beam to separate them.

Fix - extra gauges to remove the error:

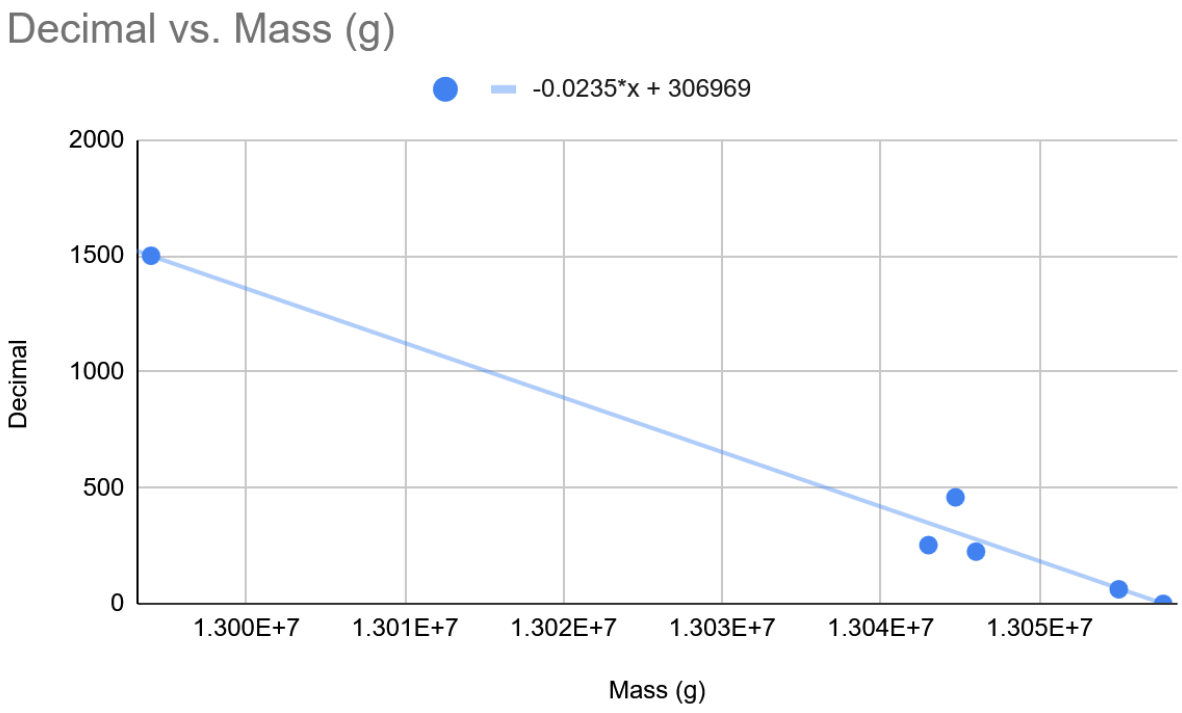
- Use a full-bridge or half-bridge configuration with gauges on both the top and bottom surfaces at the same cross-section:
    - Two gauges on opposite faces (one in tension, one in compression) form a differential pair that doubles bending sensitivity and cancels axial strain.
- Use a strain gauge rosette (three gauges at 0°, 45°, 90°) if you need to resolve multi-axial strain.
- Alternatively, put gauges symmetrically about the neutral axis and wire them into a bridge so that axial (uniform) strain cancels out while bending (opposite-sign top vs bottom) adds.
- Place gauges on both sides of the neutral axis (top and bottom) and use them as opposing legs in the Wheatstone bridge to maximize bending sensitivity and minimize axial/temperature errors.

In practice: to remove the “shortening” effect, add at least one gauge on the opposite side of the beam at the same x-location and wire them into the bridge so that axial strain cancels and bending remains.

Measurements

Object	Mass (g)	Decimal
Baseline	0	13036900
H-Bridge (Broken)	62	13032500
Weight	225	13018000
Phone (Bryans)	253	13018300
Tape Despenser	459	13013200
Thermos	1501	12980000

Slope: -0.02695711987



Pictures