

THE UNIVERSITY OF TEXAS AT ARLINGTON  
COMPUTER SCIENCE AND ENGINEERING

# MECHATRONICS LAB 6 REPORT

**ELECTROMECHANICAL SYSTEMS & SENSORS**

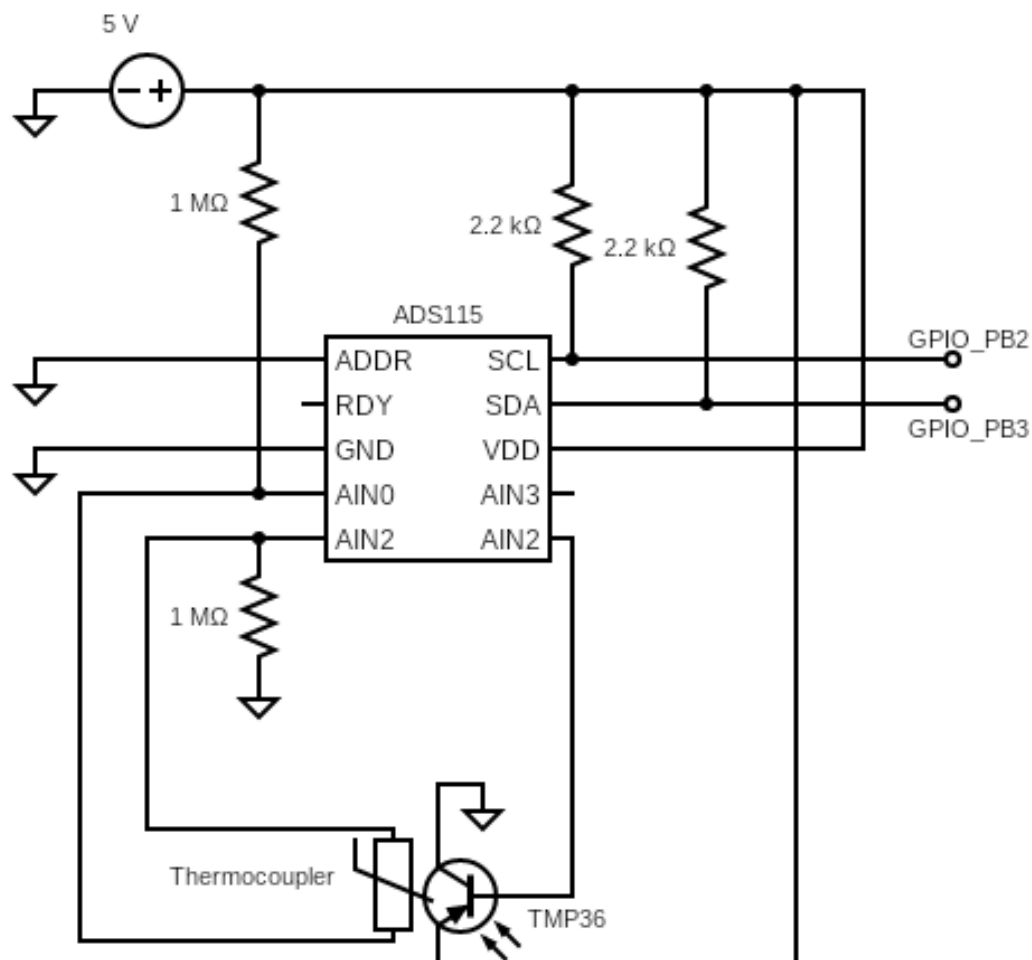
Submitted toward the partial completion of the requirements for CSE 5355-001

**Submitted by,**  
Jennifer Hernandez &  
Servando Olvera

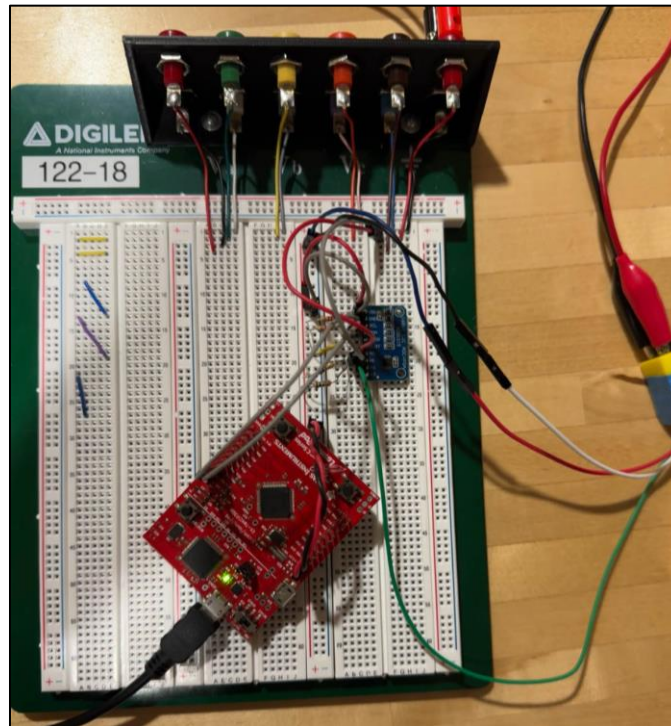
Date 11/08/2024

1. Using a type K thermocoupler jack (yellow), connect the two output wires to two inputs of an ADS1115 A/D converter so that a differential measurement can be made. To bias the inputs, tie the positive input to Vdd through a 1 M $\Omega$  resistor and tie the negative input to GND through a 1 M $\Omega$  resistor. Plug the thermocouple into the jack. Connect an TMP36 temperature sensor to one of the two remaining inputs on the A/D converter. Make sure that the temperature sensor is in contact with the jack, as the TMP36 will be used to measure the cold junction temperature of the thermocouple system. Any difference in temperature between the wires in the jack and the TMP36 and errors in the TMP36 temperature sensor will result in increased measurement error.
2. Connect the ADS1115 to the TM4C123GXL board, making sure to pull up the SDA and SCL lines with a 2k $\Omega$  resistor. Using the I2C tool, verify that you can see the ADS1115.  
The address of ADS115 was determined to be 0x48.

### Circuit Diagram



## Circuit Built



3. Write a lookup table based on NIST ITS-90 tables normalized to 0°C for the temperatures of interest (at least -80 to 300°C). Create functions to convert voltage to temperature and temperature to voltage using interpolation

## LUT Code Snippet

```
typedef struct nist_table{
    float temp;
    float voltage;
} nist_entry;

nist_entry nistTable[] = {
    {-80, -2.920}, {-70, -2.587}, {-60, -2.243}, {-50, -1.889},
    {-40, -1.527}, {-30, -1.156}, {-20, -0.778}, {-10, -0.392},
    {0, 0}, {10, 0.397}, {20, 0.798}, {30, 1.203}, {40, 1.612},
    {50, 2.023}, {60, 2.436}, {70, 2.851}, {80, 3.267}, {90, 3.682},
    {100, 4.096}, {110, 4.509}, {120, 4.920}, {130, 5.328}, {140, 5.735},
    {150, 6.138}, {160, 6.540}, {170, 6.941}, {180, 7.340}, {190, 7.739},
    {200, 8.138}, {210, 8.539}, {220, 8.940}, {230, 9.343}, {240, 9.747},
    {250, 10.153}, {260, 10.561}, {270, 10.971}, {280, 11.382}, {290, 11.795}, {300, 12.209}
};
```

**Voltage to Temperature Function:** The `voltToTemp()` function takes a voltage in mV and uses the `nistTable` to interpolate a temperature in degrees Celsius. This function is accurate with input voltages ranging from -2.920mV to 12.209mV inclusive. The temperature is calculated using the interpolation formula:

$$T = t1 + [(volt-v1)*(t2-t1)]/(v2-v1)$$

where input voltage was found to be between `v1` and `v2` (where `v1<v2`), which have values `t1` and `t2` respectively (where `t1<t2`) in the `nistTable`.

**Temperature to Voltage Function:** The `tempToVolt()` function takes a temperature in degrees Celsius and uses the `nistTable` to interpolate a voltage in mV. This function is accurate with input temperatures ranging from -80C to 300C inclusive. The voltage is calculated using the interpolation formula:

$$V = v1 + [(temp-t1)*(v2-v1)]/(t2-t1)$$

where the input temperature was found to be between `t1` and `t2` (where `t1<t2`), which have values `v1` and `v2` respectively (where `v1<v2`) in the `nistTable`.

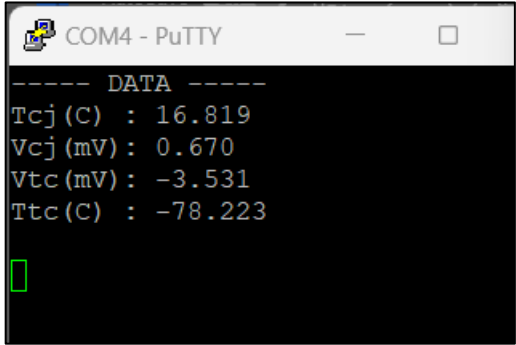
- 4. Write a function to configure the A/D and measure the TMP36 temperature. This will be the cold junction temperature (`Tcj`). Convert this cold junction temperature to equivalent thermocouple voltage (`Vcj`) using the temperature to voltage function from step 3. Note, this is called the cold junction even if the thermocouple temperature at the end of the cable is higher or lower than `Tcj`.**

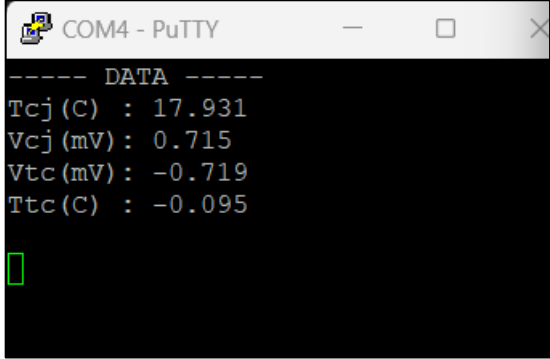
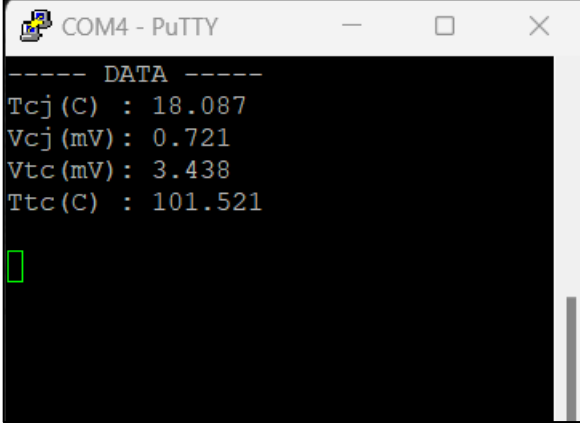
A function `getTcj()` was made to configure the ADS1115, perform a single conversion on channel AIN2, read the result, convert said result to a voltage in mV, and then calculate the temperature on the TMP36 based on the voltage. The **config** array is used to hold the specific configuration for AIN2 in reference to ground, and after starting a single conversion the Operational Status bit on the Configuration register is polled to check if conversion is complete. Once it is done, the raw ADC (**rawResult**) value is read from the Conversion register and stored. The **rawResult** value is then scaled to milli Volts using the corresponding LSB size based on the Full-Scale Range selected (**2.048 V**) for configuration. Subsequently, this value (**Vout**) is then converted to temperature using the linear conversion of 10 mV/°C and an offset of 450 mV (**Tcj**). Lastly, having calculated **Tcj**, this value is converted to mV using the function `tempToVolt()` from step 3, thus deriving **Vcj**.

5. Write a function to configure the A/D and measure the thermocouple differential voltage ( $V_{tc}$ ) using the highest PGA gain setting. Add the cold junction equivalent voltage ( $V_{cj}$ ) from step 4 to the thermocouple voltage ( $V_{tc}$ ). This now compensates for the fact that the cold junction temperature is not in a bath of ice water at  $0^{\circ}\text{C}$  (it is at room temperature – likely around  $23$  to  $25^{\circ}\text{C}$ ). Now convert the summed voltages ( $V_{cj} + V_{tc}$ ) to temperature using the tables from step 3 to get the thermocouple temperature ( $T_{tc}$ ).

A function *getVtcj()* was made to configure the ADS1115, perform a single conversion on channel AIN0 and AIN1, read the result, and convert said result to a voltage in mV. The **config** array is used to hold the specific configuration for AIN0 in reference to AIN1, and after starting a single conversion the Operational Status bit on the Configuration register is polled to check if conversion is complete. Once it is done, the raw ADC (**rawResult**) value is read from the Conversion register and stored. The **rawResult** value is then scaled to milli Volts using the corresponding LSB size based on the Full-Scale Range selected (**0.256 V**) for configuration (**Vtc**). Subsequently, having calculated **Vtc**, the cold junction equivalent voltage (**Vcj**) is added to it and the total voltage (**AddedVolt**) is then converted to temperature using the function *voltToTemp()*, deriving **Ttc**.

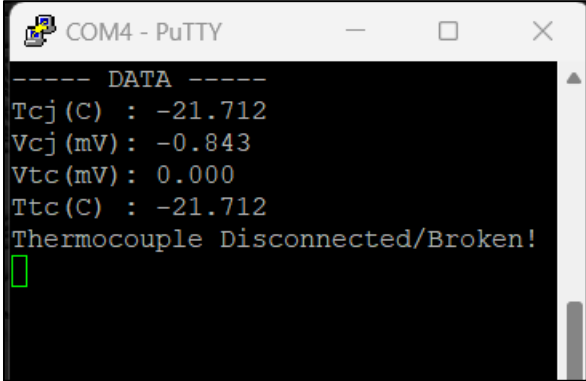
6. Verify that your solution properly calculates the temperature ( $T_{tc}$ ) when the thermocoupler is placed on dry ice ( $-78.5^{\circ}\text{C}$ ), ice water bath ( $0^{\circ}\text{C}$ ), and a soldering iron in the lab.

Placement of Thermocouple	Expected Temperature	Temperature Reading
Dry Ice	$-78.5^{\circ}\text{C}$	 <pre> COM4 - PuTTY ----- DATA ----- Tcj (C) : 16.819 Vcj (mV) : 0.670 Vtc (mV) : -3.531 Ttc (C) : -78.223 </pre>

Iced Water	$\sim 0^{\circ}\text{C}$	 <pre> COM4 - PuTTY ----- DATA ----- Tcj (C) : 17.931 Vcj (mV) : 0.715 Vtc (mV) : -0.719 Ttc (C) : -0.095 </pre>
Soldering Iron	$\sim 100^{\circ}\text{C}$	 <pre> COM4 - PuTTY ----- DATA ----- Tcj (C) : 18.087 Vcj (mV) : 0.721 Vtc (mV) : 3.438 Ttc (C) : 101.521 </pre>

**7. Add a function to take advantage of the bias network to determine whether a thermocouple is either disconnected or broken (open circuit).**

When thermocouple is disconnected the raw valued in the conversion register is 0x7FFF, which is the max positive value that can be outputted. A check is done for this value, and if it said value is read, a flag is set. When printing out the data this flag is checked and if set, “Thermocouple Disconnected/Broke” is displayed.



```

COM4 - PuTTY
----- DATA -----
Tcj (C) : -21.712
Vcj (mV) : -0.843
Vtc (mV) : 0.000
Ttc (C) : -21.712
Thermocouple Disconnected/Broke!

```