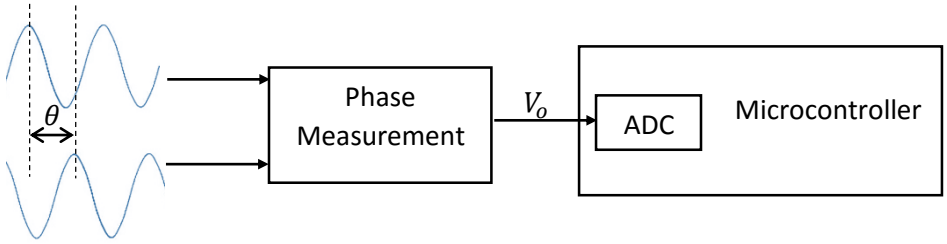
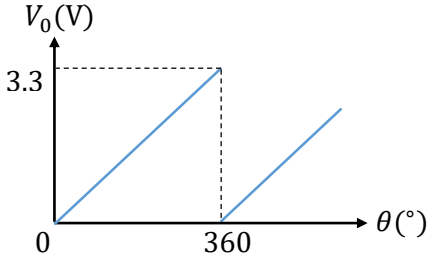
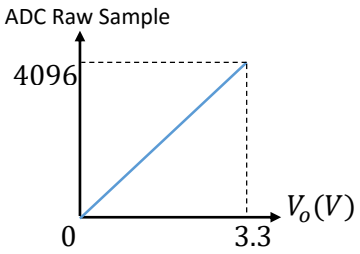
Consider the following system:




The system takes two sine waves as the inputs. The first stage (phase measurement) contains a circuit that converts the phase delay ($0° \leq \theta < 360°$) of the two sine waves into a voltage level $V_o$ ($0\text{V} \leq V_0 < 3.3\text{V}$). The second stage is a microcontroller that uses the ADC to sample $V_o$ and runs some C code to reconstruct $\theta$.

- The mapping between $\theta$ and $V_o$ is linear, as shown by the figure on the left. For example, if the two since waves have a phase delay of 180 degrees, $V_0$ will be 1.65V. The ADC is 12-bit, so the values of the raw samples are between 0 and 4095 ($0 \leq S < 4096$), representing voltages linearly scaled between 0 and 3.3V, as shown by the figure on the right.




- To average out the noise from the phase measurement, we use the ADC to collect a set of continuous samples. In our example, we collect 256 samples.

With the information above, could you implement the C code in the microcontroller that processes the collected samples from the ADC and outputs the phase delay $\theta$? Specifically:

- You need to process the raw samples to average out the noise.
- Since the code is called repeatedly and frequently, the running speed is important.
- The microcontroller natively supports 32-bit floating point operations via an FPU.
- You can assume that we have the standard C libraries.
- Be careful with the special case when $\theta$ is around zero degree, and there is noise in the input.

```
// This array stores the collected samples from the ADC
uint16_t raw_samples[256];     // The values are from 0 to 4096

/*
 * This function processes the raw samples stored in raw_samples,
 * and recovers the phase delay (theta).
 *
 * When the function is called, we assume that raw_samples
 * already has the most up-to-date samples from the ADC.
 */
float getPhaseDelayFromRawSamples() {

    // TODO: Please implement this function

    return theta;    // theta is between 0 and 360
}
```

You are given a temperature sensor to use in an application; its datasheet is linked below:
https://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf

This sensor is utilized in a system to measure ambient temperature, and controls device(s) based on the measured value.

System specifications:

- Measured temperature is expected to be between 0 and 100 °C.
- If the temperature exceeds 50 °C, a fan is switched ON using GPIO (else it is OFF).
- If the temperature drops below 5 °C, a heater is switched ON (else it is switched OFF).

Can you design a system that achieves the above? Assume:

- Any embedded platform of your choice (e.g. ARM, STM, AVR etc.).
- It is sufficient to represent the temperature value in an integer format.
- A 10-bit ADC to sample the analog voltage in to digital.
- Both the $V_{ref}$ and $V_{cc}$ of the ADC to be 5V for simplicity's sake.
- Standard C libraries.

---

```
// This variable stores latest value from temperature sensor sampled by ADC
volatile uint16_t raw_adc_sample;

/*
 * This function processes the raw ADC data and turns ON/OFF devices
 * based on the it.
 * When the function is called, we assume that raw_adc_sample
 * already has the latest sample from the ADC.
 */
uint16_t getTemperatureFromRawSample()
{
     // TODO: Please implement this function
     return degC;     // degC is between 0 to 100
}

/*
 * This function turns ON/OFF devices based on temperature.
 * as returned by getTemperatureFrom RawSample()
 */
void controlDevices()
{
     // TODO: Please implement this function
}
```

---