Analog Sensors

While computers, microcontrollers, and other devices that interface with VEX robots are digital systems, most of the real world operates as analog components, where a range of possible values exist instead of simply an arrangement of 1s and 0s. To communicate with these analog real-world systems, analog sensors like potentiometers and line trackers are used. These sensors return a number within a preset range of values in accordance with their input, as opposed to a digit sensor which simply returns an on or off state.

To take these analog inputs and convert them to information that the Cortex can actually use, ADCs (Analog to Digital Converters) are used on each of the Analog In ports to convert the analog input signals (varying voltage signals) to 12 bit integers. As a result, the range of all analog sensors when used with the Cortex is 0 to 4095 (the range of a 12 bit unsigned integer).

Initialization

Unlike many of the other VEX sensors, no initialization process is needed in the `initializeIO()` or `initialize()` functions. However, it is often worthwhile to calibrate analog sensors before using them, which would take place in the `initialize()` function. The `analogCalibrate()` function collects approximately 500 data samples over a period of half a second and returns the average value received over the sampling period. This average value can be used to account for variations like gyroscope orientation or ambient light for line trackers.

Potentiometer

Potentiometers measure angular position and can be used to determine the direction of rotation of its input. Potentiometers are best used in applications such as lifts where the sensor is not at risk of being rotated beyond its 250-degree physical constraint. Potentiometers typically do not need to be calibrated, although it may be desired as it helps account for possible shifting in the potentiometer mounting and to find the actual range of the potentiometer due to its mechanical stops as that range may be closer to 5-4090 instead of 0-4095. If the potentiometer is not calibrated, the `analogRead()` function may be used to obtain the raw input value of the potentiometer. If the sensor was calibrated, the `analogReadCalibrated()` function should be used, as it will account for the sensor's calibration and return more accurate results. The input to both of these functions is the channel number of the sensor, and an integer is returned.

Thus an example of use on a lift would look like:

```
#define POTENTIOMETER_PORT 1
#define LIFT_MOTOR 1

//while the potentiometer is not at its maximum position
while(analogRead(POTENTIOMETER_PORT) < 4095)
{
  motorSet(1,127); //activate the lift
}
```

Line Tracker

VEX Line Trackers operate by measuring the amount of light reflected to the sensor and determining the existence of lines from the difference in light reflected by the white tape and the dark tiles. The Line Trackers return a value between 0 and 405, with 0 being the lightest reading and 4095 the darkest. It is recommended that Line Trackers be calibrated to account for changes in ambient light.

An example of Line Tracker use:

main.h:

```
#define LINE_TRACKER_PORT 1
#define DRIVE_MOTOR_LEFT 1
#define DRIVE_MOTOR_RIGHT 2
```

init.c:

```
#include "main.h"

void initialize() {
  analogCalibrate(LINE_TRACKER_PORT);
}
```

opcontrol.c:

```
#include "main.h"

//2000 arbitrarily set as cutoff between light and dark
```

```
while(analogReadCalibrated(LINE_TRACKER_PORT) < 2000)
{
  // drive forward until a line is hit
  motorSet(DRIVE_MOTOR_LEFT,127);
  motorSet(DRIVE_MOTOR_RIGHT,127);
}
```

Gyroscope

One of the most powerful sensors available for the VEX Cortex is the VEX Yaw Rate Gyro. Through proper utilization of this sensors you can consistently make your robot perform precise turns.

Warning

The VEX Yaw Rate Gyro is an analog sensor which means that it is very susceptible to analog noise during its operation. When utilizing this sensor, pay special attention to the connection wires between cortex and the gyro and keep them far away from motors.

PROS provides a gyro library to simplify using it. A sample usage would be as follows:

main.h:

```
// Analog port number gyro is plugged into
#define GYRO_PORT 1

// Multiple gyros can be declared
Gyro gyro;
```

init.c:

```
void initialize(){
    // ... Other sensor initialization and port configuration
    // If gyro reads inaccurately, change "0" to desired sensitivity
    // See documentation on gyroInit() for up-to-date sensitivity details
    gyro = gyroInit(GYRO_PORT, 0);
}
```

opcontrol.c or auto.c:

```
void myFunction(){
    // ... Do work
    // Get gyro reading in degrees
    int heading = gyroGet(gyro);

    // ... Do other work
    // Reset the gyro to zero
    gyroReset(gyro);

    // ...
}
```

Accelerometer

The VEX Accelerometer measures acceleration on the x, y, and z axes simultaneously. Accelerometers can be used to infer velocity and displacement, but due to the error induced by such integration it is recommended that simply the acceleration data be used. By design of the VEX Accelerometer each axis is treated as its own analog sensors. Due to this the VEX Accelerometer requires three analog input ports on the Cortex.

Example accelerometer use:

main.h:

```
#define ACCELEROMETER_X 1
#define ACCELEROMETER_Y 2
#define ACCELEROMETER_Z 3
```

init.c:

```
#include "main.h"

void initialize() {
  analogCalibrate(ACCELEROMETER_X); //calibrates the x axis input
  analogCalibrate(ACCELEROMETER_Y); //calibrates the y axis input
  analogCalibrate(ACCELEROMETER_Z); //calibrates the z axis input
}
```

opcontrol.c:

```
#include "main.h"

//Read the acceleration data for each axis
int x_acc = analogReadCalibratedHR(ACCELEROMETER_X);
int y_acc = analogReadCalibratedHR(ACCELEROMETER_Y);
int z_acc = analogReadCalibratedHR(ACCELEROMETER_Z);
```