Sensors and PWM Control from Linux

Mike Anderson

Director of Technology

The PTR Group, LLC

Speaker/Author Details

- **Website:**
  - http://www.theptrgroup.com
- **Email:**
  - mailto:mike@theptrgroup.com
- **Linked-in:**
  - https://www.linkedin.com/in/mikeandersonptr
- **Twitter:**
  - @hungjar
- **PTR is now a subsidiary of** Huntington Ingalls Industries

PTR GROUP

THE LINUX FOUNDATION

What We Will Talk About…

- Sensors in the real world
- Interfacing to sensors via the IIO subsystem
- Analog vs. digital sensors
- Position measurement modalities
- Attitude and alignment
- IMUs
- PWM
- The Linux PWM subsystem
- Using external interfaces for PWM control

Linux and the Real World

- Linux is a major force in embedded systems and control
- In order to be a player in what was almost exclusively an RTOS world, Linux has expanded support for both input and output to physical components
- Sensors and end effectors are incredibly varied in their abilities and interfaces
  - Fortunately, especially when using the PREEMPT_RT code, Linux can handle much of what RTOSes can do

Sensors in the Real World

- Sensors are designed to help us interpret the world around us
- Probably one of the earliest sensors was the compass made from lodestone
  - Ancient Greeks found that a lodestone suspended from a string would orient itself in a north/south direction
    - Lodestone is a naturally magnetized piece of the mineral magnetite

Source: 3dbeliver.com

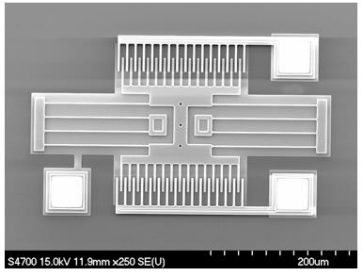
- The first evidence of using magnetized material for navigation dates back to the Song Dynasty in about 1040 CE
- From these humble beginnings, we have graduated to micron-scale transducers that are commonly found in today's sensor systems
  - Micro Electro-Mechanical Sensors (MEMS)

Source: computersmiths.com

What are we trying to Sense?

- In their simplest forms, sensors are measurement tools
  - So, even a ruler could be considered a sensor
- What we try to measure is dependent on the application
  - Distance, weight, direction, attitude relative to the Earth, rotational acceleration, time, gas or chemical PPM, the presence of radioactive particles and more
- The use of MEMS allows us to pack a lot of sensing into a small package

Source: amazon.com



Source: memnet.org

Electrical Interfaces

- Traditional interfaces to sensors would use UARTs (RS-232, RS-485, etc.) or parallel ports
  - There are still many sensor types that use UARTs, but the original PC parallel port is pretty much gone and replaced with alternatives
- Typical MEMS sensors may be GPIO bit-banged or on the I2C or SPI buses
  - Each approach has its plusses and minuses from an interface perspective
    - The biggest issue is typically speed and power consumption
- Newer systems may use interfaces based on McBSP, SPORT, MIPI, CSI-2, EPI, PPI or custom FPGA interfaces
  - Some, like CSI-2, are specific to particular sensors like cameras whereas others are more generic

Traditional Linux Sensor/Input Approaches

- Traditionally, Linux supported the **`input`** and **`hwmon`** interfaces
- The **`input`** subsystem targets human interaction devices like keyboard, joystick, mouse, drawing tablets, etc.
  - Very low interaction rates and very non-deterministic
- The **`hwmon`** subsystem was focused on low sample rate sensors (e.g., 1-2/sec) that are used for health and status monitoring of the system
  - E.g., fan speeds, humidity, voltages and system temperatures often referred to as the lm_sensors interface due to the user-space interface
    - Uses I2C/SMB or SPI but at very low data rates with little or no determinism requirements

Linux Industrial I/O Subsystem

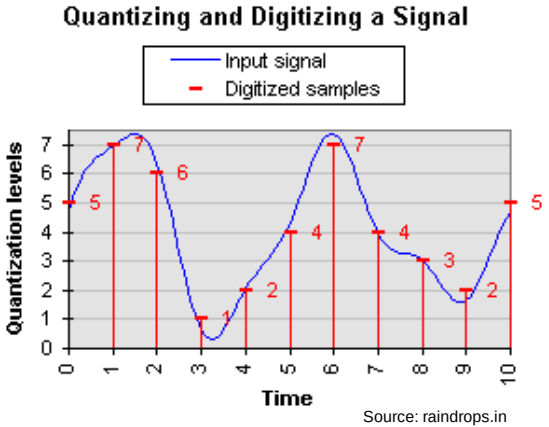
- Created in 2009 (in staging in 2.6.32), the IIO subsystem targets higher sampling rate sensors that are essentially either analog to digital converters (ADC) or digital to analog converters (DAC)
  - Over 200 drivers in the 5.x kernel series so far
- Also, includes some niche drivers that didn't fit well into the **`misc`** subsystem like clock generators and potentiometers
- Supports triggered sampling and the use of DMA transfer regions as well
  - Although, most of the sensors use KFIFO backed channels resulting in non-blocking operations

Types of Supported Sensors in IIO

- Accelerometers, gyroscopes, magnetometers and IMUs for use in motion awareness
  - Includes applications like drones and even the drop sensor in some laptops
- Temperature, barometric pressure, humidity
- Health sensors (e.g., %O2), color detection, gesture sensors and even hazardous gas and chemical detectors
- There are some great presentations that get into the details of the IIO subsystem
  - E.g., Matt Ranostay's *Industrial I/O and You* presentation from ELC 2017

Sensors Characteristics

- Sensors are typically thought of as either digital or analog
  - Analog sensors are often sampled using an analog-to-digital (A/D) in order to quantize their values into something that's easily represented in a computer
- The range of those values represents the accuracy of the A/D
  - E.g., a 12-bit A/D can represent the analog voltage as a range from 0-4095
    - There will also be a maximum sample rate (faster is better -- to a point)
- Digital sensors can be very simple, like limit switches, to more complex Inertial Measurement Units (IMUs)
- Pulse-Width Modulation allows us to simulate analog output using digital pulses
  - More on this later
- Data rates range from 100/400 KHz transfers for I2C up to 100 MHz for interfaces like SPI
- Fortunately, most μCs and small development boards like Arduinos support a wide assortment of these interfaces
  - And, the software libraries exist to make interfacing sensors relatively straightforward



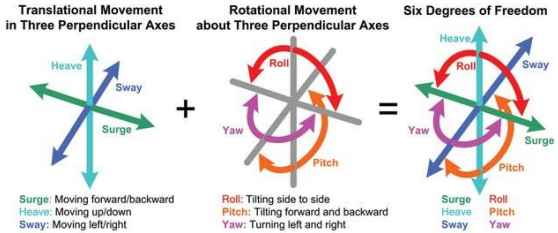
Source: raindrops.in

Absolute vs. Relative Measurements

- Much of the role of sensors is to measure quantities
- These measurements can be:
  - Absolute
    - Turn due East based on the compass
    - Speed up the wheel to 3500 RPM
  - Or Relative:
    - Turn completely around from where you are right now
    - Spin the wheel 25% faster
- However, there may need to be an absolute measurement to determine the relative change
- Whether you use absolute or relative measurements depends on the application

Degrees of Freedom

- When working with sensors, we need to understand how many different axis measurements they can provide
  - Referred to as the degrees of freedom (DoF)
- So, a device that can deliver X/Y/Z gyroscope, X/Y/Z accelerometer and X/Y/Z compass would be considered a 9 DoF sensor
- Add a 10th DoF such as altitude and you have the makings for an Inertial Measurement Unit (IMU) used in flying drones
- The more DoF, the higher the typical data rates

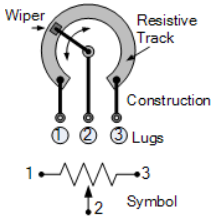


Source: hackernoon.com

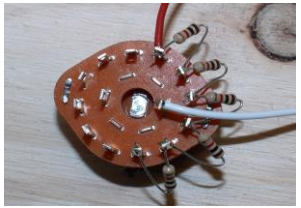
Position Sensors

- Probably the easiest of all of the position sensors is the limit switch
  - Typically implemented as a simple switch attached to actuator that indicates that you've reached some end condition or used as a tachometer
  - Switch can be N/O or N/C depending on your logic in the software
  - Often attached to a GPIO and can be several KHz if uses as a tach
- Position sensors can also take the form of a potentiometer attached to a linkage
  - Used to measure a range of motion such as the elevation of a robot arm
  - Potentiometers are generally analog sensors
- Usually, a potentiometer has 3 wires
  - VCC, GND and a signal return known as the wiper
    - You will supply the VCC and GND and measure the signal return using an A/D
- The A/D of the Arduino Uno is 10-bit (5V) and the PocketBeagle is 12-bit (1.8V or 3.3V)
  - The measured return voltage can be calibrated to represent a full range of motion
    - Be aware of the maximum voltage range and pick your potentiometer accordingly!

Source: rockwellautomation.com



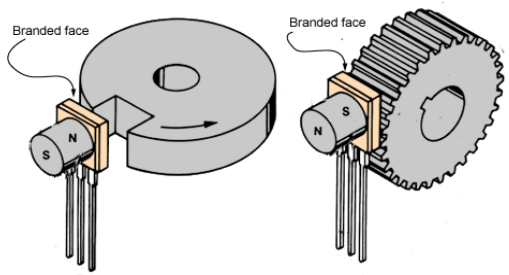
Source: electronics-tutorials.ws



Source: robotshop.com

Position/Speed Sensors

- Hall-effect sensors use metal moving through a magnetic field (induction) to indicate motion
  - Can also be used as a limit switch
- Can be used to count rotations or calculate position based on the number of gear teeth that have passed the sensor
  - This is how automotive cruise controls often work
- Can also be used as a tachometer
- Would typically be attached to a GPIO and the pulses are counted using an interrupt
  - A triggered sensor from the IIO subsystem perspective
  - Good application for the PRU of the Ti Sitara SoC, dedicated μC or an FPGA to lighten the load on the processor if the rate to be measured is high



Source: allegromicro.com

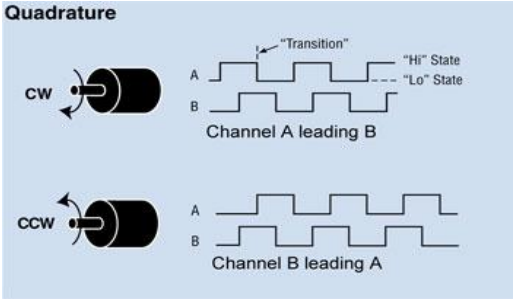
Position/Speed Sensors #2

- Another GPIO-centric position encoder is a rotary encoder
- These encoders will have a number of pulses per revolution (PPR)
  - Given the diameter of the attach point, you can determine how far the system has moved based on the number of pulses
  - Can also be used as a tachometer
- Make sure you purchase the encoder rated for the speed you're trying to measure
- Like the Hall-effect sensor, high PPR can burden the CPU

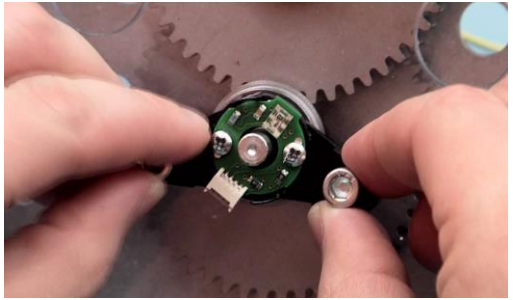
Source: grayhill.com

Position/Speed Sensors #3

- The last of the position sensors we'll talk about is the quadrature encoder
- These are similar to the rotary encoder except that you can determine forward vs. reverse motion
  - Also have a PPR rating that you need to know
    - Measuring the leading and trailing edges means 4X the PPR
- Usually a "quad shaft encoder" is attached to the axle of a wheel or motor
  - Many robotics gear boxes have specific mounting points for quadrature encoders
- Given the radius of the wheel, you know how far it's moved based on the number of pulses
- Also attached to a GPIO input
  - Some motor controllers can close the loop on quadrature encoder inputs



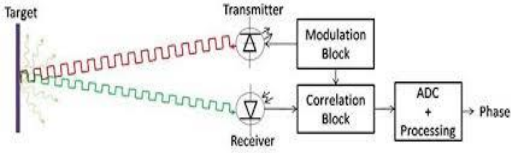
Source: dynapar.com



Source: andymark.com

Proximity Sensors

- Most distance sensors typically all work in the same general way
- A pulse goes out and you measure the amount of time it takes to return
  - Called a time-of-flight (ToF) sensor
  - Then you apply a formula to convert time into distance
- Common distance sensors use either infrared or ultrasonic modalities
  - However, laser-based sensors (lidars) are also becoming common place
- Sensor return could be analog or digital like I2C or SPI
- Beware, not all distance sensors are created equal
  - Some can range 30’ others only a few inches



Source: digikey.com

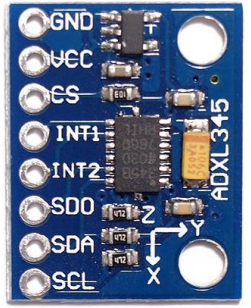
Source: sparkfun.com

Source: parallax.com

Source: robotshop.com

Measuring Acceleration

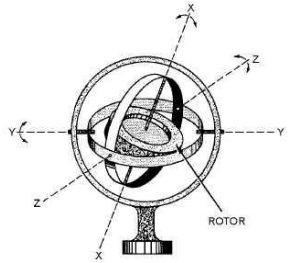
- Accelerometers measure acceleration or vibrations in terms of “G”s
  - Normal gravity at sea level is defined to be 1G
- Can be used to detect position relative to the earth
- Needs to be mounted in a given orientation to make the math needed to determine roll/pitch/yaw easier
- Can also be used to detect impacts and make adjustments to the gyro to correct for drift



Source: adafruit.com

Direction Sensors

- In order to determine the direction of something that's moving such as a drone, there are a number of possibilities
- Gyroscopes
  - Measure the *rate* of turn
  - Data can be integrated to represent your current heading relative to a given starting point
- However, gyroscopes are subject to drift and can be off by quite a bit the event of collisions
  - Use the accelerometer to know that you've had collisions



Source: electriciantraining.tpub.com

Source: xybernetics.com

Direction Sensors (2)

- Another means to determine heading is via a magnetometer (digital compass)
  - You can read an absolute measurement of direction relative to magnetic north
    - Don't forget to factor in your offset from magnetic north to true north if you need an absolute measurement on the Earth
  - But, they are subject to interference from large magnetic fields (like motors) and large pieces of iron or steel
- Like the gyro, these are typically interfaces via I2C or SPI

Source: amazon.com

Inertial Measurement Units (IMU)

- If we combine the gyro, accelerometer and magnetometer sensors together, we can create an Inertial Measurement Unit (IMU)
  - Used frequently in drones for precise navigation
- Often integrated in with a μC to handle the sampling and math from the sensors to present a simplified set of values to the user
  - This example uses an MPU-9050 9DoF sensor array and an STM32 to talk serial, SPI or I2C
- With an IMU, it's relatively easy to maintain a heading even in the wind or when getting hit by other robots
  - Add a barometric pressure sensor for a 10 DoF and you're ready to fly
- The IIO subsystem already supports several IMU drivers

Source: kauailabs.com

Pressure Sensor / Load Cell / Strain Gauge

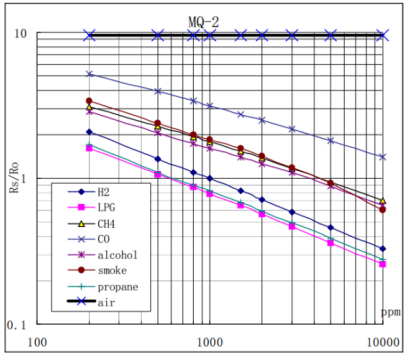
- Pneumatics/fluidics have the advantage that the actuators are very repeatable
  - However, we need to monitor pressures to maintain performance and safety
    - E.g., Automatic cut-off of compressor when pressure gets to a certain value
- Generally, these are analog sensors
- Load cells are a variant that measures weight
  - Can also measure both compression and tension or torque
- Many of these sensors use the deformation of a piezo-electric material that varies voltage based on distortion of the material
  - Voltage output is varied like a potentiometer and can be calibrated for precise measurements

Source: andymark.com

Source: sparkfun.com

Gas Sensors

- We may find ourselves needing to measure the presence of dangerous gases or substances
  - CO, CO2, LP, methane, hydrogen, isobutane, alcohol
- Most of these sensors are analog and need to be calibrated
- These often work using a small heating element
  - Make sure that you give them room to breathe
- IIO subsystem has many examples of these devices



Source: iteadstudio.com

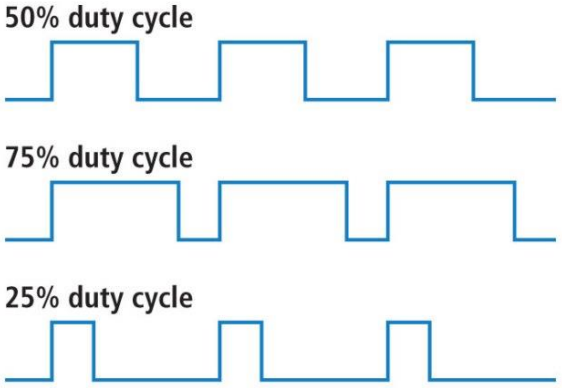
Source: bananarobotics.com

Approximating Analog Output

- So far, most of the sensors we've talked about are inputs
  - IIO does have support for DACs, but true analog outputs can be difficult to get correct
- However, we can approximate analog output using a digital circuit and a technique known as pulse width modulation (PWM)
- There are several devices that we come in contact with daily that are controlled with PWM

PWM

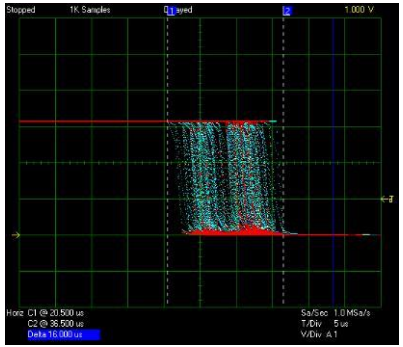
- With PWM, we are trying to approximate an analog voltage by varying the frequency and duty cycle of a digital signal
  - E.g., 50% duty cycle of a 5V system would be 2.5V
- PWM interfaces are commonly found on LEDs (like the “breathing” LED when your laptop is suspended), fans, LCD screen backlights, vibrators in cell phones, etc.
  - When you need to vary the intensity of something without a potentiometer
- However, PWM is also used with switching power supplies, stepper and servo motors



Source: Sparkfun.com

PWM Sources and Jitter

- It is certainly possible to approximate a PWM signal by bit-banging a GPIO
  - Some implementations of the Python GPIO library do this
- For many devices, the required accuracy of the PWM output is fairly low
  - LEDs and your eyes have persistence, so having some variability on the duty cycle is not noticeable

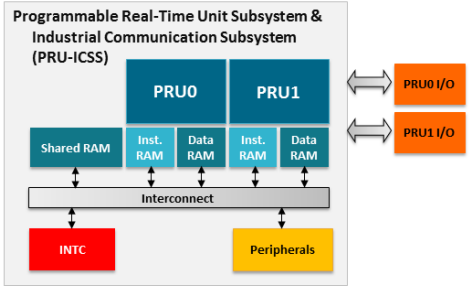


Source: microchip.com

- However, for other applications like switching power supplies, stepper motors or servos, the jitter of the duty cycle can cause disastrous results
  - The motor can literally chatter itself to death
- Linux's default preemption model and the arrival of interrupts complicate the timing of PWM signals considerably
  - Even PREEMPT_RT may not provide sufficient resolution to make certain control system function correctly

Solving the Jitter Problem

- To address both the duty cycle and frequency jitter problems and reduce CPU utilization, many SoCs now have hardware PWM generator circuits built in
  - However, there may only be one or two channels at best



Source: tii.com

- If the PWM signal is being targeted for external use such as drone motor control, then external hardware in the form of PWM controllers are likely going to be needed
  - Alternatively, auxiliary processing units like the TI Sitara's PRUs can be used to generate a solid PWM signal

Linux PWM Subsystem

- For PWM requirements such as fans and backlights, Linux has a PWM subsystem with a defined API
- Board setup code and/or device trees can be used to wire the PWM circuitry to an external pin to make it available for use
- The PWM channel can manifest itself in /dev and or /sys
  - The kernel documentation at https://www.kernel.org/doc/Documentation/pwm.txt describes the operation of the API

PWM Channels in Sysfs

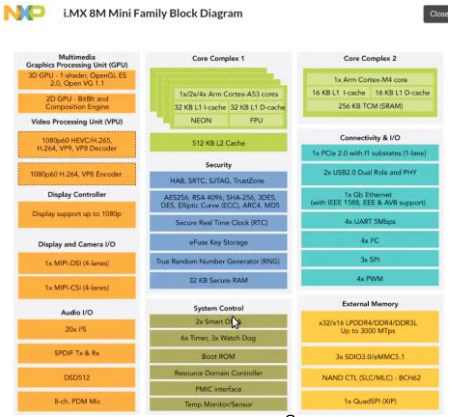
- If the CONFIG_SYSFS is enabled in the kernel, then a simple sysfs interface is provided to userspace
  - Exposed at **`/sys/class/pwm`**
    - Each PWM controller/chip is exported as **`pwmchipN`** where N is the base of the PWM chip
- Inside the sysfs directory you will find the number of PWM channels (**`npwm`** (0 relative)), and an **`export`** and **`unexport`** option for each channel
- When a PWM channel is exported, a **`pwmX`** (X is the channel number) subdirectory is created
  - Additional properties are then made visible

PWM Properties

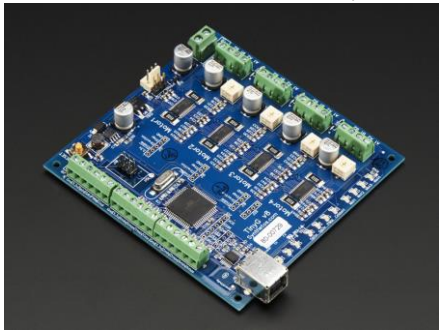
- The exported properties include:
  - **`period`** (r/w)
    - The total period of the PWM signal in nanoseconds
  - **`duty_cycle`** (r/w)
    - The active time of the PWM signal
  - **`polarity`** (r/w)
    - Changes the polarity of the signal
      - The PWM must be disabled to change this
  - **`enable`** (r/w)
    - Enable (1) or disable (0) the PWM signal
- The PWM channel will need to be connected to something via hardwiring or pin mux, etc.

Controlling Multiple External Channels

- If the application is a 3D printer, drones, etc., then there will likely be 4+ PWM channels that need to be controlled with very high accuracy
- Fortunately, there are a number of hybrid multi-core SoCs that are becoming available
  - ARM Cortex-A and Cortex-M in the same package or like the Sitara's PRU co-processors
- Additionally, there are I2C, SPI, USB and UART-based PWM controllers



Source: nxp.com


Source: adafruit.com

Summary

- There is an incredible variety when it comes to sensors
  - Fortunately, the IIO subsystem has worked examples of many of the more common types
- For many applications like backlights, the onboard PWM sources will work fine
  - Off-board devices may require the use of external hardware to provide the stability needed for precision control
- The increasing number of hybrid SoCs means that we really have a dedicated microcontroller that can handle the PWM as an attached processor
- Or, we can always use an external μC like an Arduino variant that we can command via Ethernet, USB or serial


Embedded Linux
Conference
North America