Photo of You and Your Hardware Connected

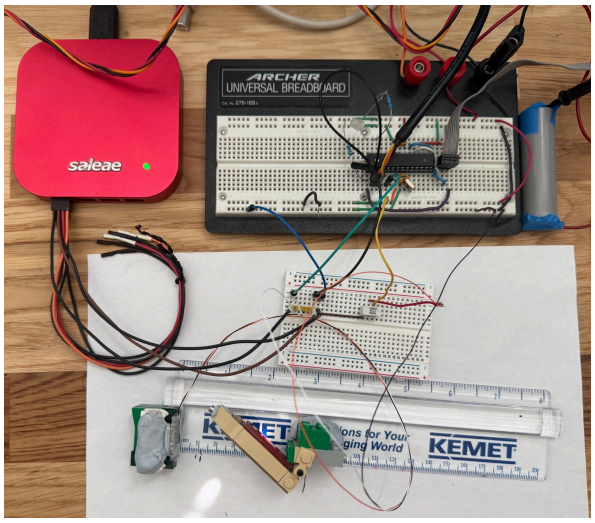




Fig 1. Hardware Setup **Fig 2. My picture**

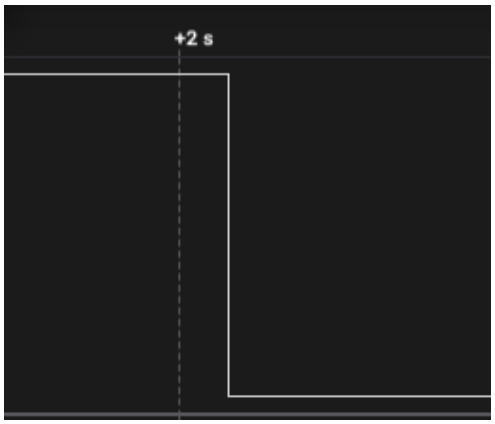
I set up an ATmega328P microcontroller with the necessary clocking and electronic components, connected a hall effect sensor, used a rod to sweep the magnet, and attached a logic analyzer to capture the UART and I/O waveforms.

Experimental Transfer Function

$$V_{\text{out}}(d) = \begin{cases} 0\text{ V (LOW)}, & d \leq 1.1\text{ cm} \\ V_{CC}\text{ (HIGH)}, & d > 1.1\text{ cm} \end{cases}$$

Tested at supply voltages of 2.7, 3.3, 3.7, 4.2, and 5.0 V

Equation 1. Transfer Function of Hall Effect Sensor



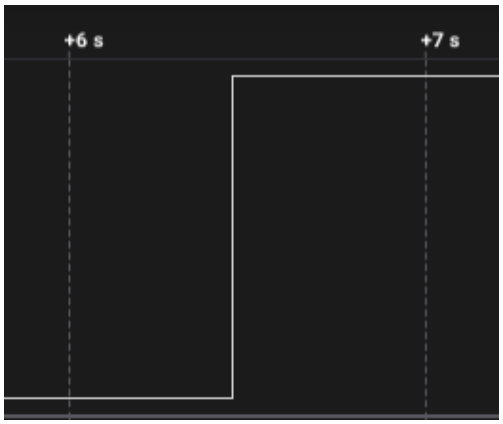


Fig 3. Magnet going from >1.1cm to 1.1cm **Fig 4. Magnet going from 1.1cm to >1.1cm**

This is a transfer function with a single trip point at d = 1.1 cm, switching between 0V (LOW) and VCC (HIGH). The trip point stays fixed at 1.1 cm across all tested voltages (2.7-5V) meaning the switching threshold is independent of the supply voltage. There is negligible hysteresis as the operating and release point both occur at 1.1 cm.

Description of Sensor Calibration

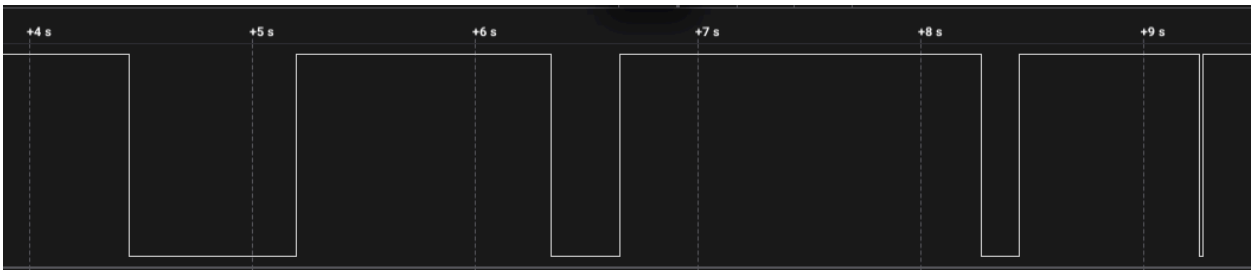


Fig 5. I/O Waveform at Various Speeds

To calibrate the hall effect sensor, a test fixture was built on a plastic ruler with the sensor mounted vertically at the 0 cm mark and a pivoting rod anchored at the 6 cm line. A permanent magnet was attached to the end of the rod at the 1.1 cm mark, allowing me to calibrate the system's maximum operating point at various speeds. The calibration procedure involved manually sweeping the rod toward and away from the sensor at low, medium, and high speeds across various supply voltages. A logic analyzer was used to monitor the sensor's output pin.

The calibration establishes 1.1 cm as the maximum operational distance between the magnet and sensor for reliable activation. This maps the physical distance to a transfer function where the output is 0V (LOW) if the distance is less than or equal to the 1.1 cm limit, and VCC (HIGH) if the distance is greater than 1.1 cm. The hall effect sensor detected the magnet cleanly with zero data loss or distortion.

Apply Statistics Concepts to the Sensor

The sensor system has a single independent variable: physical distance. While supply voltage is another variable, testing confirms it does not affect the hall effect sensor's output as long as it is within 2.7 V ≤ Vcc ≤ 5.5 V. Omitting the supply voltage variable minimizes a degree of freedom.

Testing was conducted across fifteen discrete distances ranging from 0.1 cm to 1.5 cm with 30 trials per point. The values 1.1 cm and 1.2 cm are selected as the critical boundary thresholds because they define the exact point where the binary output transitions states. Because the experimental data is perfectly binary (100% success at 1.1 cm and 0% success at 1.2 cm), traditional continuous sample variance cannot be performed. Instead, a continuous uniform probability distribution is assumed across this unmeasured 0.1 cm sampling interval. This allows any value between the last known success (a = 1.1 cm) and the first failure (b = 1.2 cm) to have an equal probability of being the true transition point. Solving for the variance quantifies the uncertainty with the unmeasured threshold gap.

$$\text{Var} = \frac{(b-a)^2}{12} = \frac{(1.2-1.1)^2}{12} \approx 0.000833 \text{ cm}^2$$

Equation 2. Variance for Hall Effect Sensor

This yields a standard deviation (SD) of 0.02886 cm.

Scheffe Band

The SD can further be used to calculate the Scheffe band. The first step is to calculate the critical value where we set p = 1 parameter, n = 450 total observations (15 points x 30 trials), n - p = 449 degrees of freedom, and = 0.05 significance level. The F-distribution value is now F(0.05, 1, 449) resulting in 3.862. Solving for the critical multiplier, M = (p x F)1/2, results in 1.965. Multiplying this by the standard deviation yields a tolerance margin of 0.057 cm. Finally, centering this margin around the 1.10 cm point produces the final confidence band equation:

$$d_{\text{band}} = 1.10 \pm 0.057 \text{ cm}$$

Equation 3. Scheffe Band for Hall Effect Sensor

This helps detect erroneous readings by defining a 95% confidence interval from 1.043 - 1.157 cm for normal mechanical operation. Because a moving magnet creates a continuously changing magnetic field, the hall effect sensor must snap cleanly between states at a specific threshold. Right at the physical edge, mechanical vibrations or electrical noise can cause the sensor to oscillate or chatter rapidly. Even though my testing yielded 100% accuracy in controlled trials, a statistical model can never be 100% definite, hence the 95% confidence interval. The Scheffe band mathematically establishes that any output flicker occurring within the 1.043 - 1.157 cm range is an expected artifact of the sensor and not a hardware failure. Consequently, any state change captured completely outside of these limits can be immediately flagged as a true operational error such as a shifted mechanical mount or hardware failure.

IOT results - data logging

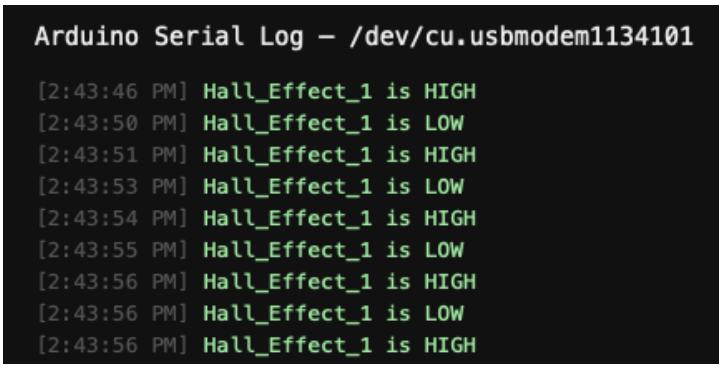


Fig 6. Python Data Logger

I wrote a C program on the ATmega328P that reads the hall effect sensor's output and sends logging data to a laptop via UART on state change. A Python program running on the laptop captures the UART signal and displays it on a Flask based web application.

The Intersection Between the Battery, the Microcontroller, and Sensor

The system implemented a direct battery-to-microcontroller-to-sensor topology avoiding any DC-DC conversion and power supply complications. The 4.2VDC battery was connected to the Vcc and

GND rails of the breadboard, allowing the ATmega328P and the hall effect sensor to be powered directly from the battery. This custom hardware layout is shown in Fig 1.

The ATmega328P microcontroller operates between 2.7V - 5.5V, while the hall effect sensor is rated to operate between 1.8V - 5.5V. The wide operating range of the microcontroller and sensor allows them to operate reliably even as the battery discharges as it is being used. Testing across several voltage levels confirmed the sensor's trip point remains immune to battery voltage droop.

What I Learned This Week

I learned how to set up the UART on the microcontroller and learned things such as the U2X setting where you set that bit to achieve double the transmission speed. This is helpful since it can reduce the error of your actual baud rate. I also learned how to set up and use a pointer to index through the UART transmit buffer byte by byte. Lastly, I learned how to use the sprintf function to format the UART transmit messages.

AI Interactions

For the C programming part, I used AI to help reinforce conceptual concepts. I am learning C better by coding myself and not allowing AI to output any code. That way, I am writing every line myself and that can make it easier to debug if there are any errors. But for the Python application, which was used to display UART messages, I utilized AI to help convert my terminal logger to a more sophisticated online logger using the Python Flask library.