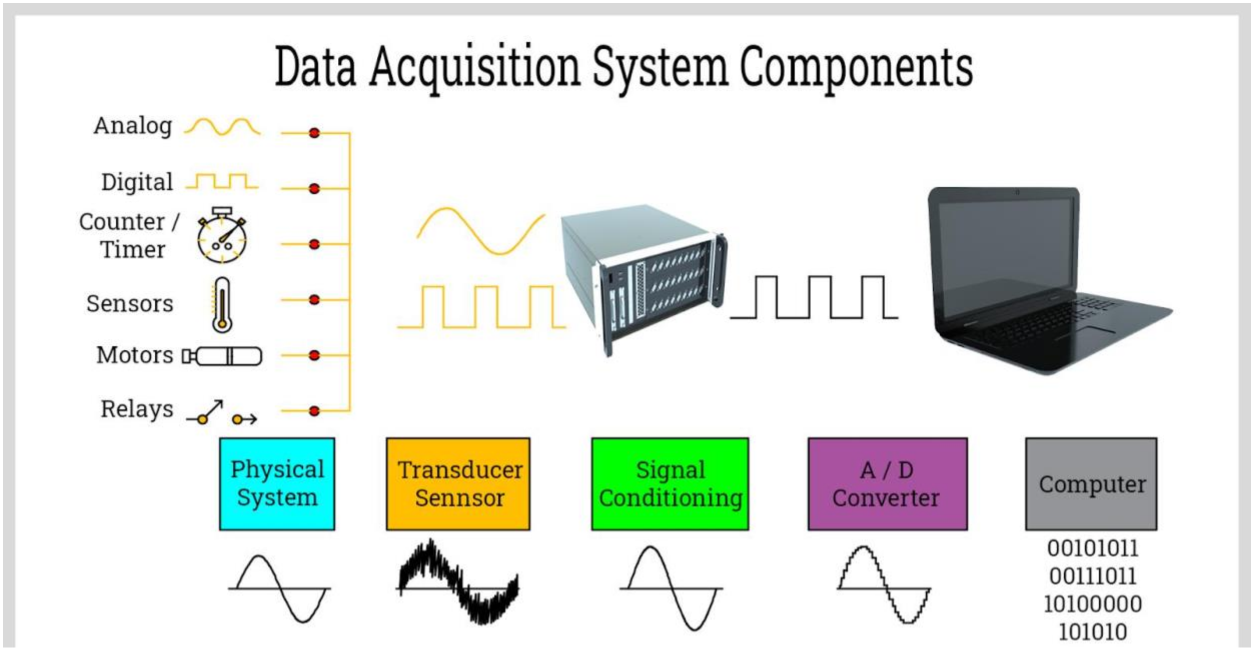
Data acquisition (DAQ) with Arduino

Data acquisition (DAQ) with Arduino involves using the microcontroller to sample real-world analog or digital signals from sensors and convert them into digital data for processing, analysis, or storage. This process is a cost-effective alternative to commercial DAQ systems.

Components of an Arduino DAQ System



A basic Arduino-based data acquisition system typically includes:

Sensors: Devices that measure physical parameters like temperature, pressure, current, or light intensity.

Signal Conditioning: Circuitry (like an op-amp amplifier) that modifies the sensor’s output signal (e.g., amplifies weak signals) so it is suitable for the Arduino’s input pins.

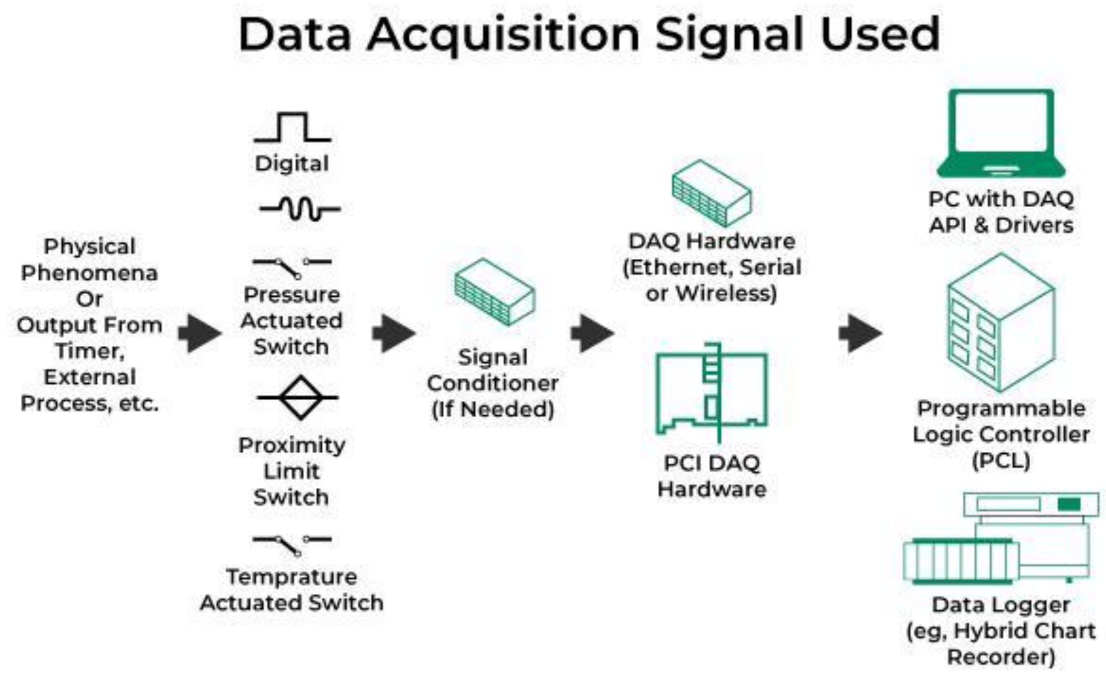
Arduino Microcontroller: The central processing unit (e.g., Arduino Uno, Nano, Mega) with a built-in Analog-to-Digital Converter (ADC) to digitize the signals.

Communication Interface: A method to transmit the digital data to a computer or storage device, most commonly via USB serial communication, I²C, or SPI.

Data Logging/Analysis Software: A program running on a connected computer (often written in Python, MATLAB, or using a special Excel macro like PLX-DAQ) to receive, timestamp, display, and log the data to a file (e.g., CSV).

Storage (Optional): An SD card shield can be used for standalone data logging without a constant PC connection.

Key Steps for Data Acquisition


Data Acquisition Signal Used
Physical Phenomena Or Output From Timer, External Process, etc.
Digital
Pressure Actuated Switch
Proximity Limit Switch
Temprature Actuated Switch
Signal Conditioner (If Needed)
DAQ Hardware (Ethernet, Serial or Wireless)
PCI DAQ Hardware
PC with DAQ API & Drivers
Programmable Logic Controller (PCL)
Data Logger (eg, Hybrid Chart Recorder)

The general process for acquiring data using an Arduino is:

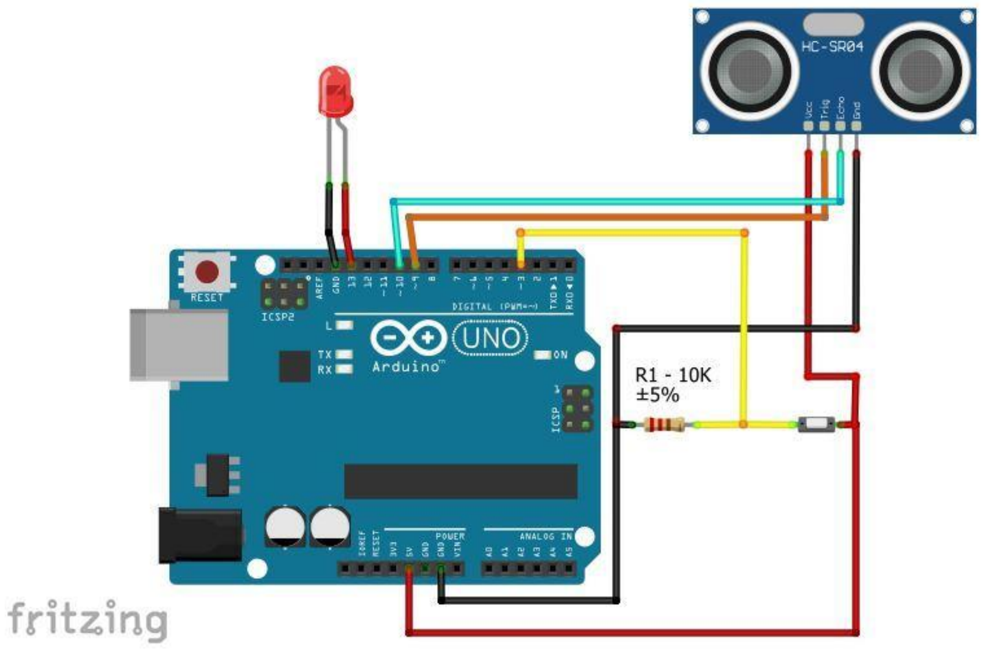
Hardware Setup: Connect your sensors to the appropriate Arduino pins (analog or digital), including any necessary signal conditioning circuitry.

Programming the Arduino (Firmware): Write an Arduino sketch in the Arduino IDE to read the sensor data and send it over a communication channel (e.g., the Serial port).

Software on the Host Computer: Run a program on your computer to open the serial port, read the incoming data, and save or display it in real-time.

Data Analysis: The collected data (often in a CSV format) can then be imported into spreadsheet software like Microsoft Excel or LibreOffice Calc for analysis and visualization.

Considerations for Performance



Resolution: Standard Arduino boards have a 10-bit ADC, which might not be sufficient for high-precision applications. External high-resolution ADC modules (16-bit, 24-bit) can be easily integrated via I²C or SPI communication protocols to improve performance.

Sampling Rate: The speed at which data can be acquired is limited by the ADC conversion time and the serial communication baud rate. Techniques like data averaging or using binary transmission can help maintain data integrity at higher speeds.

Noise Reduction: Implementing a moving average in the source code can help reduce intrinsic noise from the ADC and electronic components.