HX-1 Hybrid Rocket Engine Data Acquisition System

University of Texas at Arlington - Hybrid Rocket Division

Complete setup and usage guide for the HX-1 DAQ system.

📋 System Overview

This system reads sensor data from a hybrid rocket engine test stand and transmits it in real-time to a laptop for visualization and logging.

Architecture

```
┌─────────────────────────┐        Ethernet Cable
┌─────────────────────┐
│  Raspberry Pi 3B     │◄───────────────────────►│   Laptop/PC     │
│ (Data Acquisition)   │    TCP/IP Connection    │ (Visualization) │
│                     │                         │                 │
│ - Reads sensors via I2C │  JSON Data Packets → │ - Real-time plots │
│ - Reads thermocouple (SPI) │                 │ - CSV data logging │
│ - Controls valves (GPIO) │                   │ - Statistics    │
│ - Sends data @ 100 Hz │                      │                 │
└─────────────────────────┘          └─────────────────────┘
```

Sensors Connected

Sensor	Interface	Function	CDR Page
WIKA A-10 Pressure Transducer (x2)	I2C via ADS1115	Chamber & Tank Pressure	64
Type-K Thermocouple	SPI via MAX31855	Chamber Temperature	64
Load Cell	I2C via HX711	Thrust Measurement	64
Solenoid Valves (x4)	GPIO	Flow Control	67-68

🔧 Hardware Setup

Raspberry Pi Connections

I2C Bus (Pins 3 & 5)

```
Pin 3 (SDA) ──▶ I2C Multiplexer (TCA9548A) ──┬──▶ ADS1115 (Pressure Sensors)
Pin 5 (SCL) ──┘                              └──▶ HX711 (Load Cell)
```

I2C Device Addresses:

- TCA9548A Multiplexer: `0x70`
- ADS1115 ADC: `0x48` (default, can be 0x48-0x4B)
- HX711/Load Cell Amp: `0x2A` (depends on your adapter)

Reference: I2C addressing guide - https://learn.adafruit.com/i2c-addresses

SPI Bus (Pins 19, 21, 23, 24)

```
Pin 19 (MISO) ──┐
Pin 21 (MOSI) ──┤──▶ MAX31855 Thermocouple Amplifier
Pin 23 (SCLK) ──┤
Pin 24 (CE0)  ──┘
```

Reference: Raspberry Pi pinout - https://pinout.xyz/

GPIO Pins (Solenoid Valves)

```
GPIO 17 ──▶ Main Oxidizer Valve (via relay)
GPIO 27 ──▶ Tank Vent Valve (via relay)
GPIO 22 ──▶ Cooldown Valve (via relay)
GPIO 23 ──▶ Igniter (via relay)
```

Important: Use relay modules rated for 12V solenoids. GPIO outputs only 3.3V @ 16mA.

Pressure Sensor Wiring (4-20mA to Voltage)

WIKA A-10 pressure transducers output 4-20mA (4mA = 0 PSI, 20mA = 1000 PSI).

Convert to voltage using a 250Ω precision resistor:

```
Pressure Sensor ──▶ [+] 4-20mA [−] ──▶ 250Ω ──▶ GND
                    │
                    └──▶ ADS1115 Input
```

Voltage range: 1V (4mA) to 5V (20mA)

Reference: 4-20mA current loop tutorial - https://www.analog.com/en/analog-dialogue/articles/4-to-20-ma-current-loops.html

Ethernet Connection

Direct Connection (Pi to Laptop):

```
Raspberry Pi Ethernet Port ◀──▶ Ethernet Cable ◀──▶ Laptop Ethernet Port
```

Network Configuration:

- Raspberry Pi IP: Set static IP (e.g., `192.168.1.50`)
- Laptop IP: Set static IP (e.g., `192.168.1.100`)
- Subnet: `255.255.255.0`
- **Both must be on same subnet!**

Setting Static IP on Raspberry Pi:

bash

```
sudo nano /etc/dhcpcd.conf

# Add at end of file:
interface eth0
static ip_address=192.168.1.50/24
static routers=192.168.1.1
static domain_name_servers=8.8.8.8

# Save and reboot
sudo reboot
```

Reference: Raspberry Pi networking - https://www.raspberrypi.com/documentation/computers/configuration.html#configuring-networking

---

💻 Software Installation

On Raspberry Pi

1. Update System

```bash
sudo apt-get update
sudo apt-get upgrade -y
```

2. Enable I2C and SPI

```bash
sudo raspi-config
# Navigate to: Interface Options → I2C → Enable
# Navigate to: Interface Options → SPI → Enable
# Select Finish and reboot
sudo reboot
```

3. Verify Interfaces

```bash
ls /dev/i2c* /dev/spidev*
# Should show: /dev/i2c-1, /dev/spidev0.0, /dev/spidev0.1
```

4. Install Python Dependencies

```bash
pip3 install adafruit-blinka
pip3 install adafruit-circuitpython-ads1x15
pip3 install adafruit-circuitpython-max31855
pip3 install smbus2
pip3 install RPi.GPIO

# Or install all at once:
pip3 install -r requirements.txt
```

5. Test Hardware (I2C Device Detection)

```bash
sudo apt-get install i2c-tools
i2cdetect -y 1
# Should show addresses of connected I2C devices
```

6. Copy DAQ Script

```bash
# Copy rocket_daq_system.py to Raspberry Pi
# Edit configuration section:
nano rocket_daq_system.py
# Change LAPTOP_IP to your laptop's IP address
```

References:

- Adafruit Blinka installation: https://learn.adafruit.com/circuitpython-on-raspberrypi-linux

- I2C tools: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c

On Laptop

1. Install Python 3.7+

- Download from https://www.python.org/downloads/

2. Install Dependencies

```bash
pip install matplotlib numpy

# Or:
pip install -r requirements.txt
```

3. Find Laptop IP Address

Linux/Mac:

```bash
ifconfig
# or
ip addr
# Look for inet address under ethernet adapter (e.g., eth0 or enp0s...)
```

Windows:

```bash
ipconfig
# Look for IPv4 Address under Ethernet adapter
```

4. Copy Receiver Script

```bash
```

```
# Copy laptop_receiver.py to laptop
# No configuration needed - listens on all interfaces (0.0.0.0)
```

🚀 Running a Test

Test Procedure (CDR Page 55)

Test Parameters:

- Duration: 2 seconds per burn
- Number of burns: 3
- Interval between burns: 30 seconds (cooldown)
- Sample rate: 100 Hz
- Target chamber pressure: 440 PSI
- Expected thrust: ~153 N

Step-by-Step

1. Pre-Test Checklist

- [ ] All sensors connected and secured
- [ ] Ethernet cable connected (Pi ↔ Laptop)
- [ ] Both devices powered on
- [ ] Network connectivity verified (`ping` from Pi to Laptop)
- [ ] All valves confirmed closed
- [ ] Test stand anchored securely
- [ ] Safety equipment in place
- [ ] Personnel at safe distance (50 feet standoff - CDR page 64)

2. Start Laptop Receiver (FIRST!)

bash

```bash
python3 laptop_receiver.py
```

Expected output:

```
============================================================
HX-1 DATA RECEIVER - Host PC
============================================================

Configuration:
  Listening on: 0.0.0.0:5555
  Data directory: ./hx1_received_data

✓ Server listening on 0.0.0.0:5555
Waiting for Raspberry Pi to connect...
(Make sure Raspberry Pi script is running)
```

3. Start Raspberry Pi DAQ (SECOND)

```bash
python3 rocket_daq_system.py
```

Expected output:

```
============================================================
HX-1 HYBRID ROCKET ENGINE DATA ACQUISITION SYSTEM
University of Texas at Arlington
============================================================

Configuration:
  Laptop IP: 192.168.1.100
  Data Port: 5555
  Sample Rate: 100 Hz
  Log Directory: /home/pi/hx1_data

Press ENTER to start HX1_BURN1...
```

4. Connection Established

When connected, laptop will show:

```
✓ Raspberry Pi connected from 192.168.1.50:34567
Logging received data to: ./hx1_received_data/received_HX1_TEST_20241014_103000.csv
✓ Real-time plotting enabled

======================================================================
RECEIVING DATA
======================================================================
Press Ctrl+C to stop
```

5. Run Test

- Press ENTER on Raspberry Pi to start burn
- Watch real-time plots on laptop
- Data automatically saved to CSV on both devices
- After 2 seconds, test completes
- Wait 30 seconds for cooldown
- Repeat for burns 2 and 3

6. Post-Test

Laptop shows statistics:

```
============================================================
TEST STATISTICS
============================================================

Chamber Pressure:
  Average: 425.32 PSI
  Maximum: 449.12 PSI
  Minimum: 15.23 PSI

Tank Pressure:
  Average: 745.67 PSI
  Maximum: 798.45 PSI
  Minimum: 686.34 PSI

Thrust:
  Average: 148.56 N
  Maximum: 158.12 N
  Minimum: 0.00 N

Temperature:
  Average: 2589.34 °C
  Maximum: 2671.78 °C
  Minimum: 25.43 °C

Total Impulse: 297.12 N·s
  (Target from CDR: 1000 N·s)
============================================================
```

📁 File Descriptions

File	Purpose	Runs On
`rocket_daq_system.py`	Main DAQ script - reads sensors and transmits data	Raspberry Pi
`laptop_receiver.py`	Receives data and creates plots	Laptop
`requirements.txt`	Python package dependencies	Both
`dummy_test_data.csv`	Sample data file for testing	Reference

File	Purpose	Runs On
README.md	This file - setup instructions	Reference

Generated Files

On Raspberry Pi: /home/pi/hx1_data/

```
test_HX1_BURN1_20241014_103000.csv
test_HX1_BURN2_20241014_103045.csv
test_HX1_BURN3_20241014_103130.csv
```

On Laptop: ./hx1_received_data/

```
received_HX1_TEST_20241014_103000.csv
```

CSV Format

All CSV files use the same format:

```csv
timestamp,time_ms,chamber_pressure_psi,tank_pressure_psi,thrust_n,chamber_temp_c
2024-10-14T10:30:00.000,0,15.2,798.5,2.3,25.4
2024-10-14T10:30:00.010,10,445.8,797.8,156.9,2534.7
...
```

🐛 Troubleshooting

"Hardware libraries not available"

Cause: Adafruit libraries not installed or I2C/SPI not enabled

Solution:

```bash
```

```
# Check if I2C enabled
ls /dev/i2c-1

# If not found, enable with raspi-config
sudo raspi-config

# Reinstall libraries
pip3 install --upgrade adafruit-blinka adafruit-circuitpython-ads1x15
```

"Connection refused" or "Connection timeout"

Cause: Network configuration issue or laptop script not running

Solution:

1. Verify laptop script is running FIRST
2. Check IP addresses are correct:

```bash
# On Pi:
hostname -I

# On Laptop:
ifconfig  # or ipconfig on Windows
```

3. Test connectivity:

```bash
# From Pi:
ping 192.168.1.100  # Replace with your laptop IP
```

4. Check firewall:

```bash
```

```
# On Linux laptop:
sudo ufw allow 5555

# On Windows: Add port 5555 to firewall exceptions
```

I2C Devices Not Detected

Check wiring:

```bash
i2cdetect -y 1
```

Expected output should show device addresses (e.g., `48` for ADS1115).

If nothing appears:

- Check connections (SDA to pin 3, SCL to pin 5, ground connected)
- Verify 4.7kΩ pull-up resistors on SDA and SCL lines
- Check device power supply (most devices need 3.3V or 5V)

Reference: I2C troubleshooting - https://learn.adafruit.com/i2c-addresses/overview

"Permission denied" accessing GPIO/I2C

Solution:

```bash
# Add user to required groups
sudo usermod -a -G gpio,i2c,spi pi

# Logout and login again, or reboot
sudo reboot
```

Sample Rate Not Maintained

Symptom: Warnings like "Sample rate not maintained (loop took 15.2ms)"

Cause: System too slow or sensors taking too long to read

Solutions:

1. Reduce sample rate in configuration (e.g., 50 Hz instead of 100 Hz)
2. Close other programs on Raspberry Pi
3. Overclock Raspberry Pi (not recommended for flight hardware)

No Plot Window Appearing

Cause: matplotlib not installed

Solution:

```bash
pip install matplotlib

# On Linux, may also need:
sudo apt-get install python3-tk
```

📊 Data Analysis

Using Dummy Data for Testing

Before actual hardware tests, test the system with dummy data:

1. On Raspberry Pi:

- The script automatically uses dummy data if hardware libraries aren't available
- No need to connect sensors
- Generates realistic burn profile based on CDR specifications

2. Run both scripts normally:

```bash
```

```
# Laptop:
python3 laptop_receiver.py

# Raspberry Pi:
python3 rocket_daq_system.py
```

Data will flow and plots will update as if real sensors were connected.

Post-Processing

Use the provided CSV files for detailed analysis:

Python (Pandas):

```python
import pandas as pd
import matplotlib.pyplot as plt

# Load data
df = pd.read_csv('received_HX1_TEST_20241014_103000.csv')

# Plot thrust curve
plt.plot(df['time_ms']/1000, df['thrust_n'])
plt.xlabel('Time (s)')
plt.ylabel('Thrust (N)')
plt.title('Thrust vs Time')
plt.grid(True)
plt.show()

# Calculate total impulse (area under curve)
dt = (df['time_ms'].max() - df['time_ms'].min()) / 1000 / len(df)
total_impulse = df['thrust_n'].sum() * dt
print(f"Total Impulse: {total_impulse:.2f} N·s")
```

Excel:

1. Open CSV file in Excel
2. Select data columns

3. Insert → Chart → Scatter Plot
4. Calculate total impulse: `=SUM(thrust_n) * (time_step)`

🔒 Safety Notes

From CDR Pages 63-68:

1. **Pressure Safety:**
    - All pressurized systems designed with FOS ≥ 2
    - Hydrostatic test to 1.5 MEOP before use
    - Relief valves set to 1000 PSI
2. **Standoff Distance:**
    - Minimum 50 feet during hot fire tests
    - Use remote ignition system
    - All personnel behind protective barriers
3. **Electrical Safety:**
    - System must fail-safe (valves close on power loss)
    - All electronics in waterproof enclosure
    - Grounding required
4. **Oxygen Cleaning:**
    - All nitrous oxide system components must be oxygen cleaned
    - Follow procedure on CDR page 62-63
    - No hydrocarbon contamination allowed
5. **Emergency Shutdown:**
    - Press Ctrl+C on Raspberry Pi → all valves close immediately
    - Manual valve shutoffs accessible from safe distance
    - Fire extinguisher rated for metal fires nearby

📚 References

Documentation

- CDR Full Document (provided)
- Raspberry Pi Documentation: https://www.raspberrypi.com/documentation/
- Python Socket Programming: https://docs.python.org/3/library/socket.html

Hardware Libraries

- Adafruit Blinka: https://github.com/adafruit/Adafruit_Blinka
- ADS1115 Library: https://github.com/adafruit/Adafruit_CircuitPython_ADS1x15
- MAX31855 Library: https://github.com/adafruit/Adafruit_CircuitPython_MAX31855
- RPi.GPIO: https://sourceforge.net/projects/raspberry-gpio-python/

Tutorials

- I2C Setup: https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/i2c-clock-stretching
- SPI Setup: https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/spi
- Socket Programming: https://realpython.com/python-sockets/

📞 Support

For issues or questions:

- Review troubleshooting section above
- Check hardware connections
- Verify software configuration
- Test with dummy data first

University of Texas at Arlington - Hybrid Rocket Division