ADR-005: Arduino Uno for GPIO Safety Interlocks

Architecture Diagrams

Status: Accepted **Date:** 2025-10-15 **Deciders:** Hardware Engineer, System Architect **Technical Story:** Migration from FT232H GPIO expander to Arduino Uno

Context and Problem Statement

TOSCA v0.1-v0.8 used Adafruit FT232H USB-to-GPIO chip for safety interlock monitoring: - Footpedal deadman switch - Laser spot smoothing module health - Photodiode analog input (power verification) - Aiming beam control (via external DAC)

Problems with FT232H: - Limited to Windows/Linux (no macOS support via libftdi) - Requires FTDI driver installation (admin rights, version conflicts) - No built-in ADC (requires external ADS1115 I2C ADC for photodiode) - No hardware watchdog timer capability - GPIO expander board discontinued by Adafruit (supply chain risk)

Should we migrate to a different GPIO solution? If so, which platform?

Decision Drivers

- **Cross-platform compatibility:** Windows, Linux, macOS support
- **Hardware watchdog timer:** Independent safety layer if software hangs
- **Integrated ADC:** 10-bit minimum for photodiode monitoring (0-5V)
- **I2C support:** MCP4725 DAC control for SEMINEX aiming beam
- **Firmware customization:** Ability to implement custom safety logic
- **Supply chain stability:** Widely available, long-term production
- **Low cost:** <$30 for production units
- **USB serial simplicity:** Avoid driver installation issues

Considered Options

- Arduino Uno (ATmega328P, USB serial, built-in ADC, I2C, programmable)
- Raspberry Pi Pico (RP2040, USB serial, ADC, I2C, programmable)
- Teensy 4.0 (ARM Cortex-M7, fast, programmable, but less common)

- Keep FT232H (continue with existing solution)
- Industrial GPIO controller (Modbus, expensive, overkill)

Decision Outcome

Chosen option: “Arduino Uno”, because it provides cross-platform USB serial communication, integrated 10-bit ADC, I2C support, and programmable firmware for custom watchdog timer. Widely available, mature ecosystem, and proven reliability. Migration completed October 2025.

Positive Consequences

- **Cross-Platform:** Works on Windows/Linux/macOS without driver installation (CDC serial driver built into OS)
- **Hardware Watchdog:** Custom firmware implements 1000ms timeout with 500ms heartbeat requirement
- **Integrated ADC:** 10-bit ADC on A0 for photodiode monitoring (0-5V, ~5mV resolution)
- **I2C Built-In:** A4/A5 pins for MCP4725 DAC (SEMINEX aiming beam control)
- **Programmable Safety Logic:** Custom firmware can implement complex interlock logic
- **Supply Chain:** Arduino Uno is ubiquitous, long-term availability guaranteed
- **Low Cost:** ~$25 for official Arduino Uno, ~$5 for clones
- **Mature Ecosystem:** Extensive documentation, libraries, community support

Negative Consequences

- **Firmware Development:** Requires custom Arduino sketch (C++ firmware)
- **Limited Processing:** ATmega328P is 16 MHz 8-bit (sufficient for GPIO but not complex processing)
- **USB Serial Latency:** ~1ms latency vs. FT232H direct GPIO (acceptable for 100Hz polling)
- **Manual Protocol:** Custom serial protocol required (no standardized command set)

Pros and Cons of the Options

Arduino Uno (ATmega328P)

- Good, because cross-platform USB serial (CDC driver built into OS)
- Good, because 10-bit ADC on A0-A5 (photodiode monitoring without external ADC)
- Good, because I2C on A4/A5 (MCP4725 DAC for aiming beam)
- Good, because programmable firmware (custom watchdog timer, safety logic)
- Good, because widely available, long-term production guaranteed
- Good, because mature ecosystem (Arduino IDE, libraries, examples)
- Good, because low cost (~$25 official, ~$5 clone)
- Bad, because requires custom firmware development (Arduino C++)
- Bad, because USB serial latency (~1ms vs. direct GPIO)
- Bad, because 16 MHz 8-bit processor (limited processing power)

Raspberry Pi Pico (RP2040)

- Good, because faster processor (ARM Cortex-M0+ dual-core @ 133 MHz)
- Good, because 12-bit ADC (better resolution than Arduino)
- Good, because USB serial built-in
- Good, because I2C support
- Good, because low cost (~$4)
- Bad, because newer platform (less mature ecosystem than Arduino)
- Bad, because requires custom firmware (MicroPython or C/C++)
- Bad, because no hardware watchdog timer by default (software implementation)
- Bad, because less familiar to team (Arduino more widely known)

Teensy 4.0 (ARM Cortex-M7)

- Good, because very fast (600 MHz ARM Cortex-M7)
- Good, because 12-bit ADC
- Good, because USB serial, I2C support
- Good, because Arduino-compatible IDE
- Bad, because less common (supply chain risk compared to Arduino)
- Bad, because higher cost (~$25-30)
- Bad, because overkill for GPIO monitoring (600 MHz for 100Hz polling)

Keep FT232H

- Good, because already implemented and working
- Good, because direct GPIO access (low latency)
- Bad, because Windows/Linux only (no macOS support)
- Bad, because requires FTDI driver installation
- Bad, because no built-in ADC (requires external ADS1115)
- Bad, because no hardware watchdog capability
- Bad, because Adafruit board discontinued (supply chain risk)

Industrial GPIO Controller

- Good, because designed for industrial safety applications
- Good, because Modbus/EtherCAT protocols (standardized)
- Good, because galvanic isolation (noise immunity)
- Bad, because very expensive ($200-500+)
- Bad, because overkill for desktop medical device
- Bad, because complex configuration and integration

Links

- Arduino Uno Documentation
- ATmega328P Datasheet
- TOSCA Watchdog Firmware
- Related: [ADR-006-hardware-watchdog-timer.md]

Implementation Notes

Migration Completed: October 2025 (v0.9.x)

Firmware: `arduino_watchdog_v2.ino` - **Heartbeat Protocol:** Control software must send "H" every 500ms - **Watchdog Timeout:** 1000ms (laser disabled if heartbeat missed) - **Serial Protocol:** Custom ASCII commands at 115200 baud - `H` - Heartbeat (reset watchdog timer) - `Rxxx` - Read GPIO/ADC (e.g., `RA0` reads photodiode) - `Dxx:y` - Write digital pin (e.g., `D02:1` motor on, `D02:0` motor off) - `Axxx` - Write analog DAC (e.g., `A2048` sets aiming beam to 50% via MCP4725) - **Safety Interlocks:** - D2 (Output): Smoothing motor control - D3 (Input): Vibration sensor (smoothing module health) - A0 (ADC): Photodiode voltage (0-5V, 10-bit → 0-1023) - A4/A5 (I2C): MCP4725 DAC for SEMINEX aiming beam (12-bit, 0-4095)

Python Integration (`gpio_controller.py`):

```
import serial

class GPIOController:
    def __init__(self, port='COM4', baud=115200):
        self.serial = serial.Serial(port, baud, timeout=0.1)
        self.watchdog_active = False

    def send_heartbeat(self):
```

```
        """Send watchdog heartbeat (call every 500ms)"""
        self.serial.write(b'H\n')

    def read_photodiode(self) -> float:
        """Read photodiode voltage (0-5V)"""
        self.serial.write(b'RA0\n')
        response = self.serial.readline().decode().strip()
        adc_value = int(response)  # 0-1023
        voltage = (adc_value / 1023.0) * 5.0
        return voltage

    def set_smoothing_motor(self, enabled: bool):
        """Control smoothing motor"""
        cmd = b'D02:1\n' if enabled else b'D02:0\n'
        self.serial.write(cmd)

    def set_aiming_beam(self, dac_value: int):
        """Set aiming beam brightness (0-4095 via MCP4725)"""
        cmd = f'A{dac_value:04d}\n'.encode()
        self.serial.write(cmd)
```

Watchdog Heartbeat Pattern:

```
from PyQt6.QtCore import QTimer

class SafetyWatchdog:
    def __init__(self, gpio_controller):
        self.gpio = gpio_controller
        self.timer = QTimer()
        self.timer.timeout.connect(self._send_heartbeat)
        self.timer.start(500)  # 500ms interval

    def _send_heartbeat(self):
        self.gpio.send_heartbeat()
```

Hardware Connections: - Arduino Uno USB → Windows PC (COM4) - D2 → Smoothing motor driver (5V logic) - D3 ← MPU6050 vibration sensor (I2C interrupt pin) - A0 ← Photodiode amplifier circuit (0-5V) - A4/A5 → MCP4725 DAC (I2C, address 0x62) → LDD200 driver → SEMINEX aiming beam

Benefits Realized: 1. **Cross-Platform Confirmed:** Tested on Windows 10, Windows 11, Ubuntu 22.04 (WSL), macOS Monterey 2. **Watchdog Timer Working:** Laser auto-shutoff confirmed when heartbeat stops 3. **Photodiode Accurate:** 10-bit ADC provides ~5mV resolution (sufficient for 0-5W laser monitoring) 4. **Aiming Beam Control:** MCP4725 12-bit DAC allows precise aiming beam brightness adjustment 5. **Supply Chain Secure:** Arduino Uno available from 50+ distributors worldwide

Review History

Date	Reviewer	Notes
2025-10-15	Hardware Engineer	Arduino Uno selected, firmware development started
2025-10-20	System Architect	Migration complete, testing successful
2025-11-04	Documentation Review	Formalized as ADR-005

Template Version: 1.0 **Last Updated:** 2025-11-04