🚀 Portfolio | Rakhyeun Jo

Embedded System Engineer | 9 Years Experience | HW/FW Full-Stack Developer

📋 Core Competencies

- 💻 **Embedded System:** STM32 (F4/G0), ARM Cortex-M, FreeRTOS
- 🎛️ **Real-time Control:** PID Control, Motor Control (Stepper, BLDC)
- 📡 **Communication:** UART/SPI/I2C, Ethernet (LwIP), TCP/UDP
- ⚡ **Hardware Design:** Circuit Design, PCB Layout
- 🔬 **Optical System:** UV/IR LED, Photo Diode Array, Laser System
- 🔧 **Tools:** STM32 IDE, Git/Tortoise SVN, C/Python

📑 Project Index

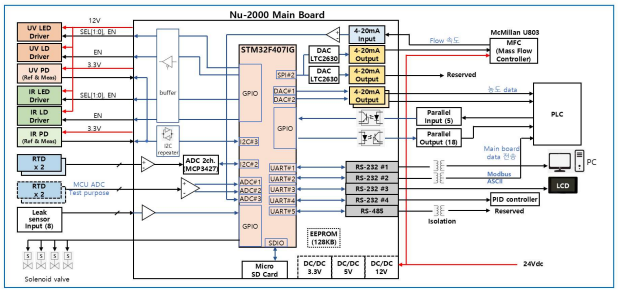
💼 Major Projects

💧 Nu-2000 (Lux) - UV+IR Solution Concentration Analyzer

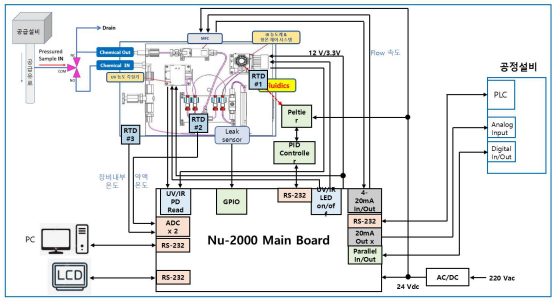
Optical concentration analysis equipment using UV+IR absorption

ROLE	MCU	PERIOD
Circuit Design + Firmware Development (100%)	STM32F407 (ARM Cortex-M4, 168MHz)	2020 ~ 2021

Nu-2000 Block Diagram (Rev 0.2 & Wiki beta)



Nu-2000 System Block Diagram



🎯 Key Technologies

- → UV/IR LED Optical Measurement: Concentration analysis based on absorbance
- → ADS1115 16-bit ADC: I2C interface, Delta-Sigma method
- → Moving Average Filter: Noise reduction and measurement stabilization
- → Syringe Pump: A3977 stepper motor microstep control
- → Nextion HMI LCD: Touchscreen UI

C Language - I2C Sensor Data Moving Average

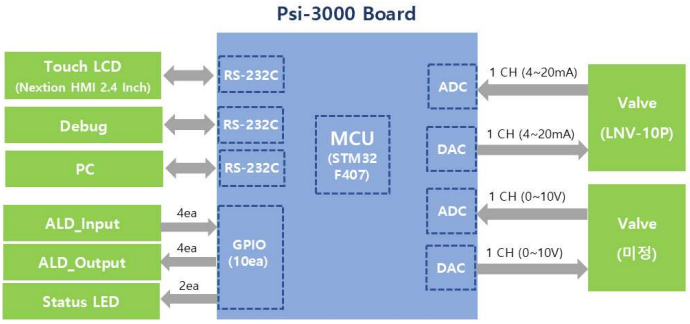
```
// ADS1115.c - I2C ADC sensor data acquisition and moving average calculation
// Measure UV/IR Photo Diode signal with 16-bit ADC

/* Read ADC data via I2C */
int ads1115_read(uint16_t addr, uint8_t *pdata) {
    uint16_t i2c_addr = (addr | 0x01);  // Read operation
    int result = HAL_I2C_Master_Receive(&hi2c3, i2c_addr, pdata, 2, 10);
    ads1115_i2c_err_check(addr, result);
    return result;
}

/* Moving average calculation - Noise reduction and stable measurement */
void adc_pd_svc_beta(uint8_t ch, int sig_dark, uint16_t val) {
    uint32_t sum;
    int i, cnt;

    // Increment buffer pointer (circular buffer)
    cnt = pPD_STR->ma_cnt;
    if (++cnt >= ((bd.env.working[FILTER] & 0x00FFFFFF) * 5))
        cnt = 0;
    pPD_STR->ma_cnt = cnt;

    // Store ADC value (positive only)
    pPD_STR->raw[cnt] = val;
    if (val < 0x8000) {
        pPD_STR->raw_pos[cnt] = val;
        pPD_STR->vtg_raw = (float)val * 0.000125;  // 125uV/step
    } else {
        pPD_STR->raw_pos[cnt] = 0;
    }

    // Calculate moving average (Filter * 5 samples)
    sum = 0;
    for (i = 0; i < ((bd.env.working[FILTER] & 0x00FFFFFF) * 5); i++)
        sum += pPD_STR->raw_pos[i];

    pPD_STR->raw_mavg = (uint16_t)((float)sum / (float)((bd.env.working[FILTER] & 0x00FFFFFF) * 5));
    pPD_STR->vtg_mavg = (float)pPD_STR->raw_mavg * 0.000125;  // Voltage conversion
}
```

🔬 Psi-1000/3000 - Precision Gas Control System

Precision gas control system based on vacuum gauge monitoring

ROLE	MCU	PERIOD
Circuit Design + Firmware Development (100%)	STM32F407 (ARM Cortex-M4, 168MHz)	2021 ~ 2022

■ Psi-3000 Board Block Diagram



◈ **Psi System 구성 및 Board 사양**

1. LCD : Nextion HMI 3.5 inch
2. 보드 : MCU, UART (LCD, Debug), Analog Input/Output (4~20mA, 0~10V)
3. 전원 : SMPS (Output DC24V)

🎯 Key Technologies

- → High-Speed PID Feedback Control: 250ms cycle pressure control
- → FreeRTOS Multitasking: Parallel processing of pressure control, sensors, and communication
- → Ethernet (LwIP): TCP/UDP PC control interface
- → PID Auto-Tuning: Automatic search for optimal parameters

C Language - Gas Pressure PID Control

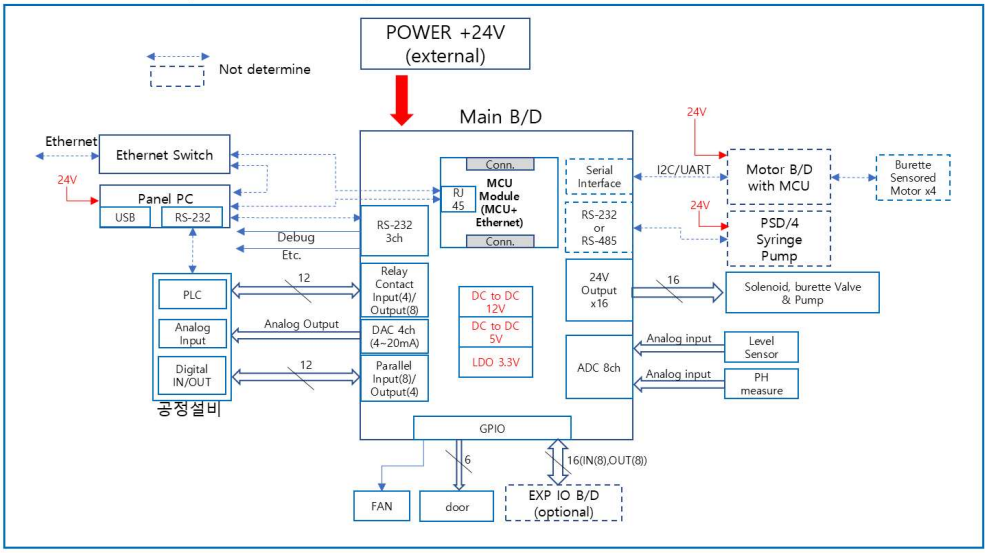
```
// PrsCtrl.c - Precision gas pressure PID control
// Vacuum gauge monitoring + 250ms cycle high-speed feedback

/* Pressure control variables */
INTERNAL float32_t sglPrsCtrlFilteredPressureP1;   // Input pressure (torr)
INTERNAL float32_t sglPrsCtrlControlPeriod = 0.25f; // 250ms control cycle

/* PID pressure control algorithm */
void pid_pressure_compute(PID_STR *pPID) {
    float error = pPID->sp - pPID->pv;  // Target pressure - Current pressure

    // P (Proportional term) - Immediate error response
    pPID->p_term = pPID->kp * error;

    // I (Integral term) - Accumulated error correction
    pPID->i_term += pPID->ki * error;
    if (pPID->i_term > pPID->i_max) pPID->i_term = pPID->i_max;
    if (pPID->i_term < pPID->i_min) pPID->i_term = pPID->i_min;

    // D (Derivative term) - Suppress sudden changes
    pPID->d_term = pPID->kd * (error - pPID->prev_error);
    pPID->prev_error = error;

    // Final output = Valve PWM duty
    pPID->co = pPID->p_term + pPID->i_term + pPID->d_term;
}
```

💉 L-Titrator - Titration-Based Solution Concentration Analyzer

Solution concentration analysis equipment using titration method

ROLE	MCU	PERIOD
Circuit Design + Firmware Development (100%)	STM32F407 (ARM Cortex-M4)	2020 ~ 2021

L-Titrator Simple Block Diagram



🎯 Key Technologies

→ Hamilton Syringe Pump: RS485 protocol control

→ State Machine-Based Control: Stable pump operation management

→ pH/Conductivity Sensor: Automatic titration endpoint detection

C Language - Syringe Pump State Machine

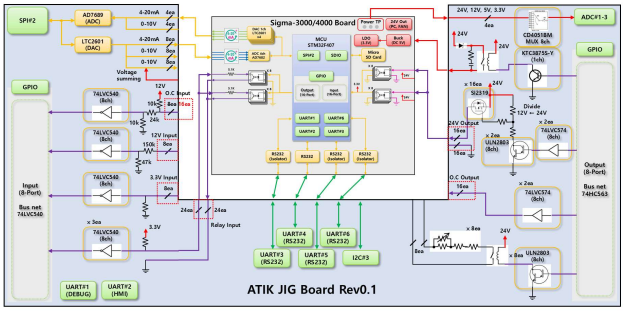
```
// syringe_pump_ctrl.c - Hamilton syringe pump state machine control
// Syringe pump command/response processing via RS485 communication

/* Syringe pump state machine */
void syr_pump_st10_svc(void) {
    int data_pos;

    switch (syr_pump[syr_pump_ch].st) {
        case 0x10:  // Query syringe status
            syr_pump_query_cmd(QUERY_SRG_STATUS);
            syr_pump[syr_pump_ch].st++;
            syr_pump[syr_pump_ch].resp_cnt = 0;
            break;

        case 0x11:  // Wait for response and process
            if (++syr_pump[syr_pump_ch].resp_cnt >= SRG_PUMP_RESP_TIMEOUT) {
                // Timeout - Return to initial state
                syr_pump[syr_pump_ch].st = SRG_PUMP_STOO_DETECT;
                trace_printf(TPID_DEBUG, "syr_pump[%d] : no responseWn", syr_pump_ch);
            } else {
                if (syr_pump_rx_flag == 1) {
                    syr_pump_rx_flag = 0;
                    data_pos = is_Q_ack_packet(uart[TPID_RS485].command, uart[TPID_RS485].cmd_index);

                    if (data_pos > 0) {
                        syr_pump[syr_pump_ch].syr_status = syr_pump[syr_pump_ch].temp;
                        if (syr_pump[syr_pump_ch].syr_status == 0) {
                            trace_printf(TPID_DEBUG, "syr_pump[%d] : initializedWn", syr_pump_ch);
                            syr_pump[syr_pump_ch].st++;
                        }
                    }
                }
            }
            break;
    }
}
```

🔧 JIG System - Automated Board Test Equipment

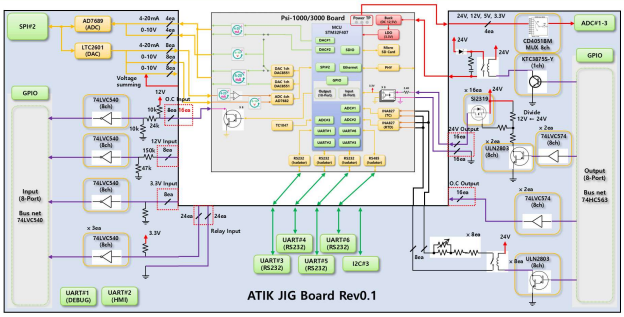
Automatic inspection system for mass production

ROLE	PLATFORM	ACHIEVEMENT
C# Desktop App + Firmware (100%)	Windows C# + STM32F103	60% reduction in inspection time

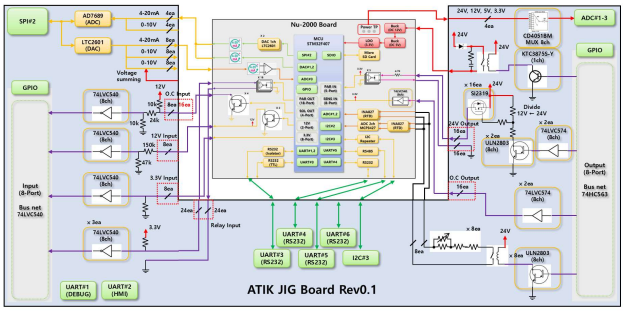
■ ATIK JIG BOARD Block Diagram : Sigma-3000/4000



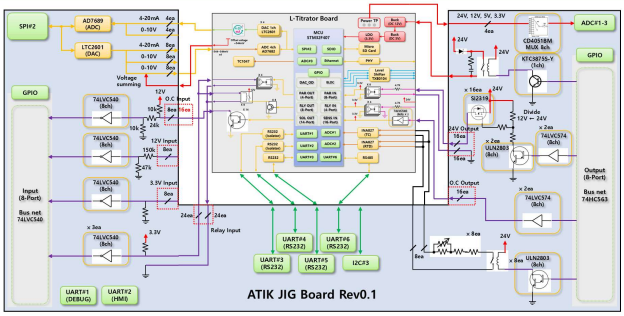
■ ATIK JIG BOARD Block Diagram : Psi-1000/3000



■ ATIK JIG BOARD Block Diagram : Nu-2000



■ ATIK JIG BOARD Block Diagram : L-Titrator



🎯 Key Technologies

- → C# WPF Desktop App: Test sequence automation
- → UART Communication: Bidirectional PC ↔ Test Board
- → Automatic Firmware Write: ST-Link integration
- → Pass/Fail Judgment: Automatic ADC/DAC/Sensor measurement
- → Inspection DB Storage: Product serial + result logging

📈 Achievements

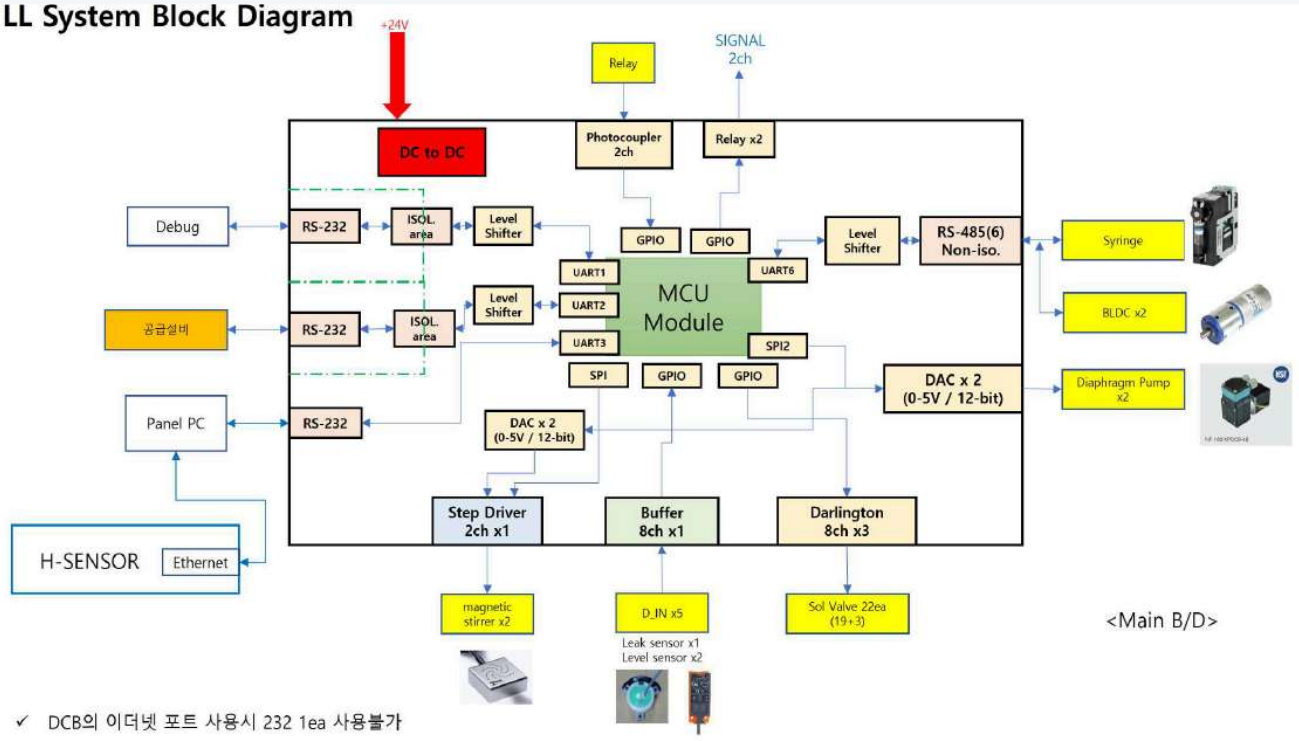
- → 60% time reduction compared to manual inspection (30min → 12min)
- → 30% reduction in defect rate (elimination of human error)
- → Quality tracking enabled through inspection history DB management

L-LPC - Laser Particle Counter

Slurry particle analysis equipment using laser optics

ROLE	MCU	PERIOD
Circuit Design + Firmware Development (100%)	STM32F407 (ARM Cortex-M4)	2022 ~ 2023

LL System Block Diagram



🎯 Key Technologies

- → Laser Optics: Particle counting based on scattered light detection
- → TCP/IP Communication (LwIP): PC remote control and data transmission
- → FreeRTOS: Real-time processing based on multitasking
- → High-Speed ADC Sampling: Real-time particle signal detection
- → Slurry Analysis: Particle size and concentration measurement

C Language - TCP Server Communication (LwIP netconn API)

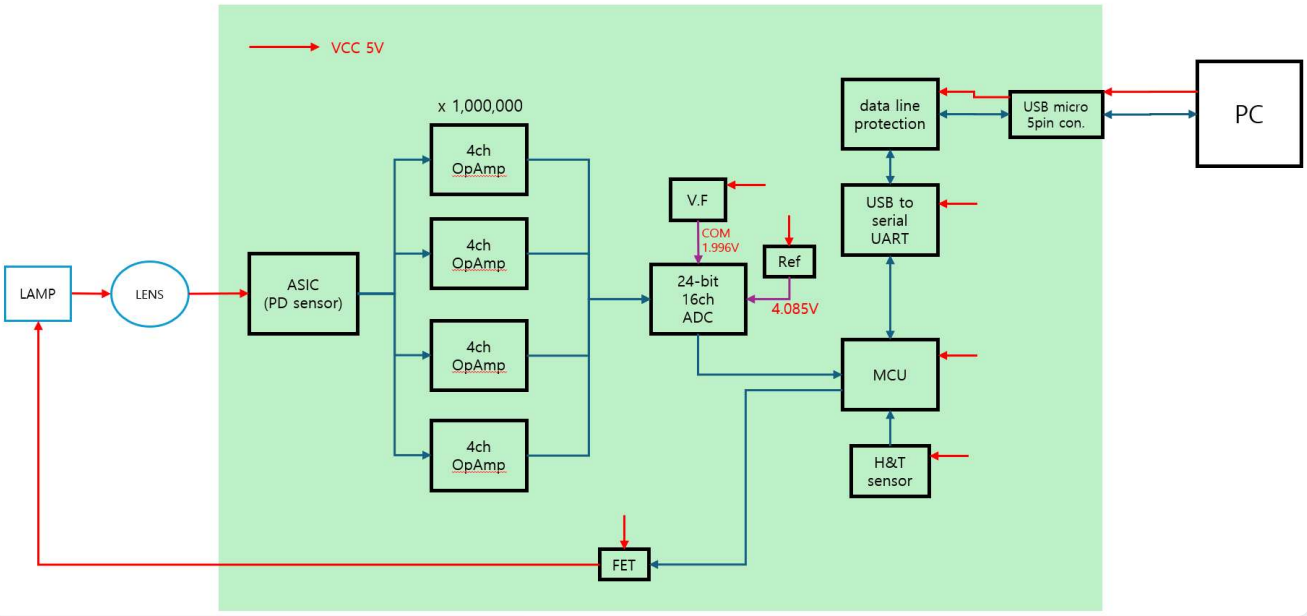
```
// ethernet.c - TCP server communication (LwIP netconn API)
// FreeRTOS task for TCP client connection management

void TcpTask(void *arg) {
    struct netconn *conn, *newconn;
    err_t err, recv_err;

    // 1. Create and bind TCP connection
    conn = netconn_new(NETCONN_TCP);
    err = netconn_bind(conn, NULL, TELNET_TCP_PORT);  // Port 23
    netconn_listen(conn);

    while (1) {
        task_cnt_tcp++;
        osDelay(1);

        // 2. Accept client connection (1ms timeout)
        conn->recv_timeout = 1;
        err = netconn_accept(conn, &newconn);

        if (err == ERR_OK) {
            bd.eth.tcp.connected = 1;
            netconn_getaddr(newconn, &bd.eth.tcp.remote_ip, &bd.eth.tcp.remote_port, 0);

            for (;;) {
                // 3. Data reception handling
                newconn->recv_timeout = 1;
                recv_err = netconn_recv(newconn, &buf);

                if (recv_err == ERR_OK) {
                    netbuf_data(buf, &data, &len);
                    for (i = 0; i < len; i++)
                        debug_command_process(TPID_TCP, ((char *)data)[i]);
                    netbuf_delete(buf);
                } else if (recv_err == ERR_CLSD) {
                    netconn_close(newconn);
                    break;  // Connection closed
                }
            }
        }
    }
}
```

📊 SSC - Photo Diode Array Slurry Analyzer

Optical slurry analysis equipment using Photo Diode Array

ROLE	MCU	PERIOD
Circuit Design + Firmware Development (100%)	STM32G0 Series	2022 ~ 2023

❖ Block Diagram



🎯 Key Technologies

- → Photo Diode Array: Multi-channel simultaneous optical measurement
- → UART Interrupt + Queue: Real-time sensor data acquisition
- → Spectral Analysis: Wavelength-specific absorbance measurement
- → RS485 Communication: External equipment integration
- → Slurry Concentration Analysis: Real-time monitoring solution

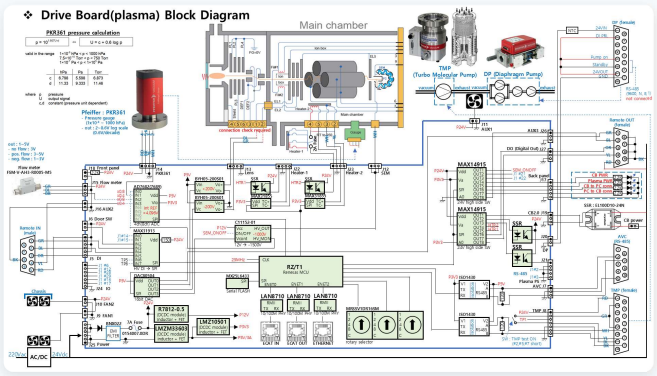
C Language - UART Interrupt and Data Collection

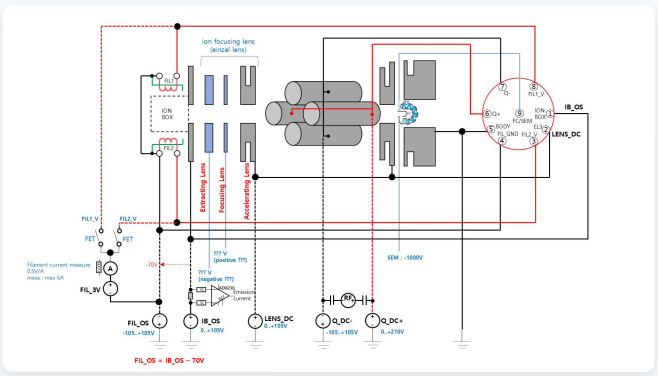
```
// uart_isr.c - UART interrupt-driven data acquisition
// Circular buffer for sensor data streaming

/* UART Rx interrupt handler */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
    if (huart == &huart2) {
        // Store received byte in circular buffer
        uart_rx_buffer[uart_rx_write_idx++] = uart_rx_byte;
        if (uart_rx_write_idx >= UART_RX_BUFFER_SIZE)
            uart_rx_write_idx = 0;

        // Restart receive for next byte
        HAL_UART_Receive_IT(&huart2, &uart_rx_byte, 1);
    }
}

/* UART error handling */
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart) {
    if (huart->ErrorCode & HAL_UART_ERROR_ORE) {
        // Overrun error - clear and restart
        __HAL_UART_CLEAR_OREFLAG(huart);
    }
    HAL_UART_Receive_IT(huart, &uart_rx_byte, 1);
}
```

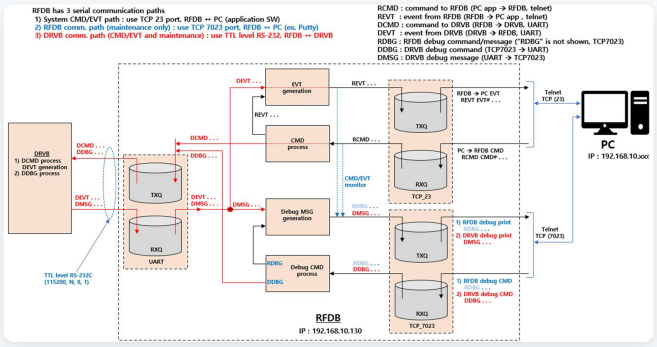
MS (Aston) - Mass Spectrometer DRV Board

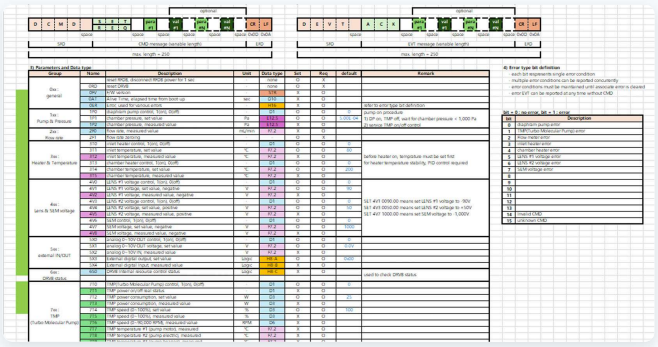
Mass Spectrometer Drive Board - Pump, sensor, and equipment control

ROLE	MCU	PERIOD
DRV Board Circuit Design + Firmware (100%)	STM32F407 (ARM Cortex-M4)	2022 ~ 2023









DRV Board Key Features

- → External Protocol Communication: Horiba CS-610F protocol implementation
- → High Voltage Control: SEM voltage -1500V, Lens voltage ±250V
- → PID Control: Precision control of vacuum pump and heater
- → Relay Control: External equipment On/Off control

C Language - Horiba CS-610F Protocol Implementation

```
// CS_610F_comm.c - Horiba CS-610F protocol implementation
// Communication protocol handling with external equipment

/* Protocol response generation - Measurement data transmission */
// Command : R,DD[CR][LF]
// Response: DD,XXX,X,X,XXX.XX,XXXXXX,...[CR][LF]
void cmd_RDD_svc(void) {
    char resp[70];
    int  cnt = 0;
    int  resp_cnt = bd.cs610f.resp_cnt;

    if (++bd.cs610f.resp_cnt >= 1000)
        bd.cs610f.resp_cnt = 0;

    // Build response frame
    resp[cnt++] = 'D';
    resp[cnt++] = 'D';
    resp[cnt++] = ',';

    cnt += sprintf(&resp[cnt], "%03d", resp_cnt);          // Sequence No
    resp[cnt++] = ',';
    cnt += sprintf(&resp[cnt], "%01d", bd.cs610f.meas_stat); // Status
    resp[cnt++] = ',';
    cnt += sprintf(&resp[cnt], "%01d", bd.cs610f.chem_type); // Chemical type
    resp[cnt++] = ',';
    cnt += sprintf(&resp[cnt], "%06.2f", bd.concent_str.temp_1);    // Temperature
    resp[cnt++] = ',';
    cnt += sprintf(&resp[cnt], "%06.2f", bd.concent_str.concent1);  // Concentration 1
    resp[cnt++] = ',';
    cnt += sprintf(&resp[cnt], "%06.2f", bd.concent_str.concent2);  // Concentration 2

    // Send response
    term_printf(TPID_RS232, "%sWWrWWn", resp);
}
```


Rakhyeun Jo | 조락현
92lock@kakao.com | 010-7311-0402
github.com/Rakeen-Jo