TECHNOLOGICAL UNIVERSITY OF THE PHILIPPINES

ELECTRONICS ENGINEERING DEPARTMENT

# ASSIGNMENT NO. 2

CPET11 – BET-CPET-3A / 7:00 AM – 10:00 AM W

Submitted By:

Arenas, Joseph

Estrada, Cyruz Adrienne

Gutierrez, Geo Kentzer

Pacis, Lian Gil

Recaña, Jordan

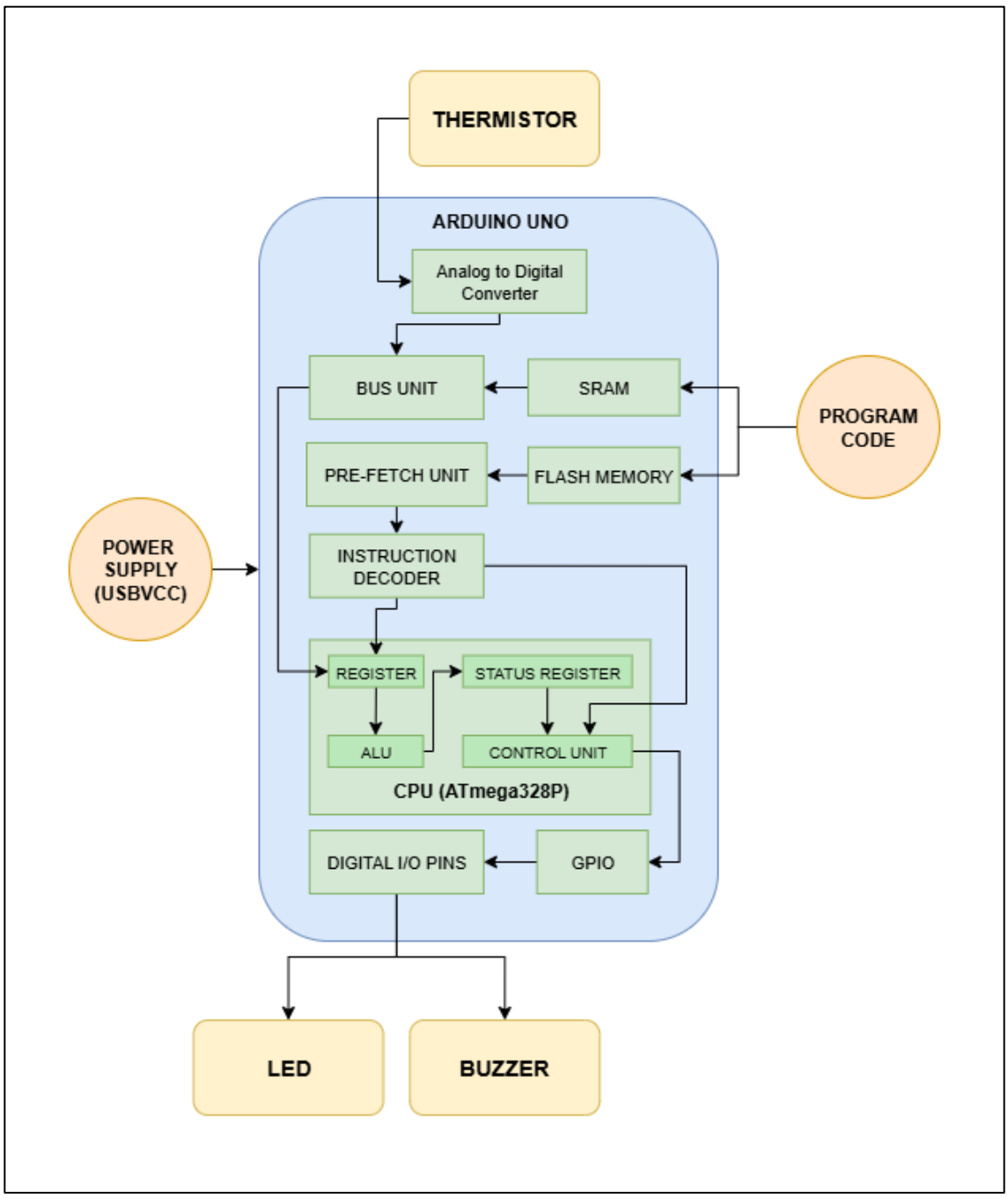
Submitted To:

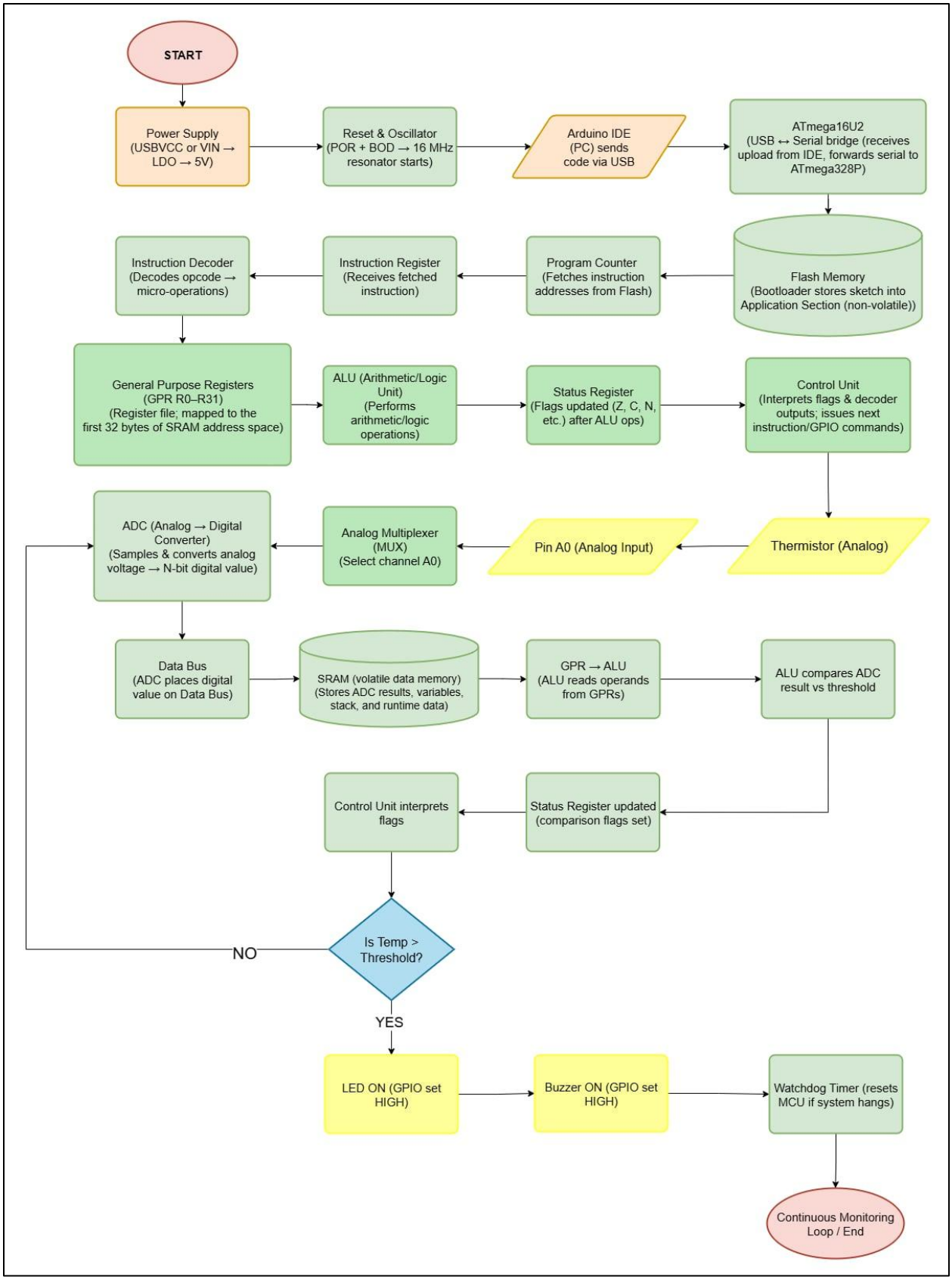
Prof. Ralph Sherwin Corpuz

Faculty

As group, work on the following requirements:

1. In reference to your module 1 requirement (i.e., IPO model), design a block diagram (with flowcharts) that showcases the different parts/components/modules/registers/other system peripherals that elaborate the relationship of all inputs and outputs. Think beyond the box. (25 points)




START
Power Supply (USBVCC or VIN → LDO → 5V)
Reset & Oscillator (POR + BOD → 16 MHz resonator starts)
Arduino IDE (PC) sends code via USB
ATmega16U2 (USB ↔ Serial bridge (receives upload from IDE, forwards serial to ATmega328P)
Flash Memory (Bootloader stores sketch into Application Section (non-volatile))
Program Counter (Fetches instruction addresses from Flash)
Instruction Register (Receives fetched instruction)
Instruction Decoder (Decodes opcode → micro-operations)
General Purpose Registers (GPR R0–R31) (Register file; mapped to the first 32 bytes of SRAM address space)
ALU (Arithmetic/Logic Unit) (Performs arithmetic/logic operations)
Status Register (Flags updated (Z, C, N, etc.) after ALU ops)
Control Unit (Interprets flags & decoder outputs; issues next instruction/GPIO commands)
Thermistor (Analog)
Pin A0 (Analog Input)
Analog Multiplexer (MUX) (Select channel A0)
ADC (Analog → Digital Converter) (Samples & converts analog voltage → N-bit digital value)
Data Bus (ADC places digital value on Data Bus)
SRAM (volatile data memory) (Stores ADC results, variables, stack, and runtime data)
GPR → ALU (ALU reads operands from GPRs)
ALU compares ADC result vs threshold
Status Register updated (comparison flags set)
Control Unit interprets flags
Is Temp > Threshold?
NO
YES
LED ON (GPIO set HIGH)
Buzzer ON (GPIO set HIGH)
Watchdog Timer (resets MCU if system hangs)
Continuous Monitoring Loop / End

2. Assuming you are using Atmel ATMega328P MCU for the IPO model process module, map out the program and data memories including their address locations that you will use to characterize the overall operation of the system. Specify the architecture to use. (25 points)

In program memory, it stores the main program code including the heat detection logic, thermistor reading algorithm, alarm conditions, and control instructions for buzzer/LED.

**Size:** 32 KB (32,768 bytes) of Flash memory
**Addresses:** 0x0000 – 0x7FFF

| PROGRAM MEMORY MAPPING | | |
|---|---|---|
| **ADDRESS RANGE** | **SIZE ALLOCATION** | **CONTENT/ PURPOSE** |
| **Application Flash Section** | | |
| 0x0000 – 0x01FF | 512 bytes (0.5 KB) | • **INITIALIZATION**<br>- Initialization for set-up, configuring ADC for the thermistor input, and initializing the I/O pins for buzzer and LED |
| 0x0200 – 0x03FF | 512bytes (0.5 KB) | • **SENSOR READING**<br>- Sensor reading for ADC conversion from thermistor, and it stores raw digital values |
| 0x0400 – 0x05FF | 512 bytes (0.5 KB) | • **TEMPERATURE CONVERSION**<br>- Convert ADC value to temperature |
| 0x0600 – 0x07FF | 512 bytes (0.5 KB) | • **DECISION LOGIC**<br>- This is where decision logic will occur, comparing the temperature with threshold. Example if the threshold is 50 degrees Celsius, it will set an |

| | | |
|---|---|---|
| | | alarm flag if exceeded with the said threshold. |
| 0x0800 – 0x09FF | 512 bytes (0.5 KB) | • **ALARM CONTROL**<br>- Alarm control where the LED and Buzzer will turn on if an overheat is detected, while turn off when the temperature is safe. |
| 0x0A00 – 0x6FFF | 24 KB (24576 bytes) | • **MAIN LOOP OF APPLICATION**<br>- Main loop where continuous monitoring occurs such as continuous sensor reading, checking and updating outputs. |
| **Boot Flash Section** | | |
| 0x7000 – 0x7FFF | 4 KB (4096 bytes) | • **BOOTLOADER SECTION**<br>- Bootloader for uploading and updating sketches in Arduino.<br>- Saves the last system state or settings, values from the calibrated thermistor, and the set temperature threshold. |

The data memory of the Arduino Uno R3 (SRAM) temporarily holds the different values to be noted from the program code implemented by the microcontroller as well as the values and signals harnessed by the various components in this circuit.

**Size:** 2KB (2048 bytes)
**Addresses:** 0x0000 - 0x07FF

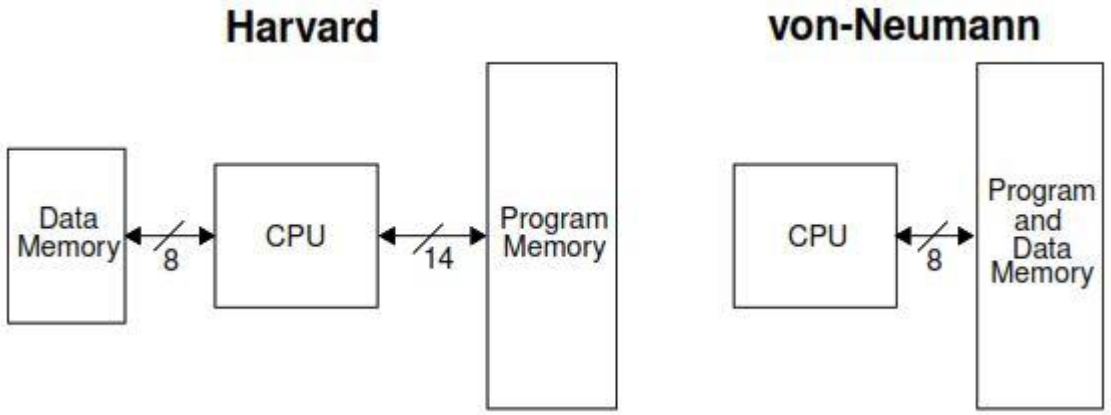
| DATA MEMORY MAPPING | | |
|---|---|---|
| **ADDRESS RANGE** | **SIZE ALLOCATION** | **CONTENT/ PURPOSE** |
| 0x0000 - 0x001F | 32 bytes | • **GPR (R0-R31)**<br>- Holds the temporary values to be stored in the General Purpose Registers of the ATMega328p. |
| 0x0020 - 0x005F | 64 bytes | • **I/O Registers**<br>- Configures, initializes and stores the values of the I/O components utilized by the microcontroller. |
| 0x0060 - 0x00FF | 160 bytes | • **Extended Peripherals**<br>- Stores the values taken by the peripheral components of the microcontroller, most importantly the Analog-to-Digital Converter (ADC) peripheral component of the Arduino Uno. |
| 0x0100 - 0x0101 | 2 bytes | • **ADC_RAW (uint16)**<br>- 16-bit value that stores the analog reading of the thermistor in this specific memory address in the SRAM for the program's future reference. |
| 0x0102 - 0x0103 | 2 bytes | • **TEMP_X10 (int16)**<br>- Actual temperature value stored in SRAM after ADC_RAW is taken into 0x0400 – 0x05FF in the Flash |

| | | |
|---|---|---|
| | | Program Memory for Temperature Conversion. |
| 0x0104 - 0x0105 | 2 bytes | • **THRESH_X10 (int16)**<br>- Takes the value that determines the temperature limit of the circuit stored in the program memory, specifically in 0x0600 – 0x07FF and stores it in this particular memory address in SRAM for future reference. |
| 0x0106 | 1 byte | • **ALARM_FLAG (uint8)**<br>- Acquired by flash program memory address 0x0800 – 0x09FF which then directs 0x0020 - 0x005F of the SRAM in order to direct the indicator to give off the alarm when THRESH_X10 is met. |
| 0x0107 | 1 byte | • **LED_STATE**<br>- Records the state (HIGH or LOW) of the LED for the program's reference. |
| 0x0108 | 1 byte | • **BUZZER_STATE**<br>- Records the state (HIGH or LOW) of the buzzer for the program's reference. |
| 0x0109–0x010A | 2 bytes | • **SAMPLE_MS**<br>- Value taken from 0x0A00 – 0x6FFF of the Flash Program Memory that tells how often the SRAM would monitor the thermistor found in memory addresses 0x0020 - 0x005F as they house the actual address |

| | | |
|---|---|---|
| | | of the pin the thermistor utilizes. |
| 0x010B–0x010E | 4 bytes | • **LAST_MS**<br>- Tracker of the last time the SRAM monitored the thermistor reading. |
| 0x010F–0x0110 | 2 bytes | • **ADC_CAL_OFF (int)**<br>- Utilized by the flash program memory address 0x0400 – 0x05FF in order to convert ADC_RAW from memory address 0x0100 - 0x0101 of the SRAM. |
| 0x0111–0x0112 | 2 bytes | • **RESERVED**<br>- Reserved for future use of the program. |
| 0x0113–0x01FF | 237 bytes | • **BUFFER**<br>- Further free space for future variables or logs specifically for the program's use. |
| 0x0200–0x07FF | 1.5kb | • **STACK/HEAP**<br>- Remaining memory of the SRAM. Stack is also stored in these memory addresses. |

Referring from Figure 8 of Module 2 - Microcontroller Overview, the Atmel ATMega328P utilizes Harvard architecture where it allows separate program and data memories, providing faster execution of instructions, security and efficient performance in less-demanding embedded systems, as these type of systems demand speed and simplicity. Additionally, Harvard Architecture consumes less power, reducing wasted cycles—a very important factor to eliminate for battery-powered embedded systems. And because our system must read changing sensor values, decide quickly and notify right away, the ATMega328P's Harvard architecture would make a great fit for the system. Due to the architecture's feature of separated program and data memories, the MCU can acquire

instructions and access sensor data simultaneously, decreasing latency and keeping response time predictable. In short, the Harvard architecture makes the ATmega328P a good fit for the system because it helps the microcontroller run faster, use less power, and quickly respond to sensor changes."




**References:**


- Atmel. (n.d.). *ATmega328/P*. https://docs.arduino.cc/resources/datasheets/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf


3. Specify the percentage contribution of each group member (100% total). Each member should sign to confirm the breakdown of contribution.

Contributions of each group member:

ARENAS, JOSEPH ..... 20%

ESTRADA, CYRUS ..... 20%

GUTIERREZ, GEO KENTZER ..... 20%

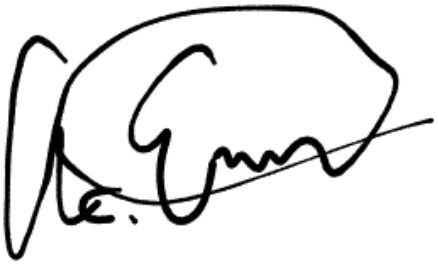
PACIS, LIAN GIL ..... 20%

RECAÑA, JORDAN ..... 20%

**TOTAL** ..... 100%

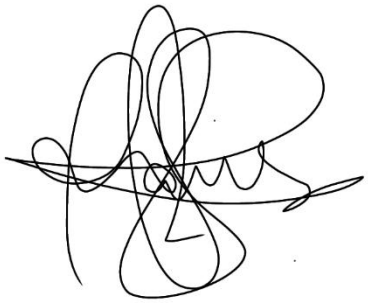
Signature of each group member:

ARENAS, JOSEPH



ESTRADA, CYRUZ

GUTIERREZ, GEO KENTZER



PACIS, LIAN GIL

RECAÑA, JORDAN