Definitions

Acronyms	Meaning
CLI	Command-Line Interface
PID	Proportional integral derivative
SPI	Serial Peripheral Interface
UART	Universal asynchronous receiver-transmitter

EDMMS Temperature Controller

Macallister Armstrong
Jeremy Evans
Anthony Kirkland
Lorand Mezei

Team Members

Jeremy Evans	Lorand Mezei	Anthony Kirkland	Macallister Armstrong
			

• CS Senior
• Software developer
• **Contributions:**
  ○ PID
  ○ SPI
	• CS Senior
	• Mathematics Tutor
	• **Contributions:**
	  ○ PID Algorithm Research
		• CS Senior
		• DBA
		• **Contributions:**
		  ○ Project management
		  ○ CLI
		  ○ Data structures
			• CS Senior
			• ERP Business Systems Analyst, J. Rettenmaier, USA
			• **Contributions:**
			  ○ Project Management
			  ○ CLI
			  ○ SPI

Acknowledgements

- Client: Allin Kahrl
- Machine Shop Specialist
- Engineering Design, Manufacturing, and Management Systems
- Western Michigan University

- Faculty Advisor: Colin MacCreery
- Faculty Specialist I
- Department of Computer Science
- Western Michigan University

Agenda

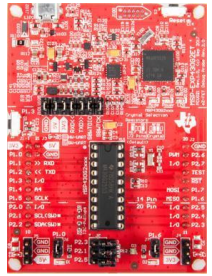
Introduction | Methodology | Results | Demo | Conclusion

- Introduction
  - Overview of specifications
  - Overview of hardware
  - Need for project
- Methodology
  - CLI
  - Data structures
  - SPI
  - PID algorithm
- Results
  - Problems encountered
  - Items produced
- Demo
- Conclusion
  - Lessons learned
  - Recommendations for future development

Overview of hardware

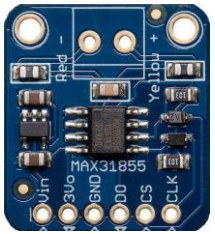
PID: TI MSP-EXP430G2ET

- Development board for the MSP430G2553 microcontroller.
- On-Board EZ-FET flash emulator for deployment and communication via UART.



Measurement: MI MAX31855 Thermocouple-to-Digital Converter

- Digitizes the signal from a K-type thermocouple and outputs a 32-bit packet of data. We extracted the top 14 bits to obtain the temperature.
  - Resolves temperatures to 0.25°C.
- Communicates with the MSP430 via SPI.



40A HOYMK Solid State Relay

- Sends a 40A current to generate heat when a signal is inputted from the MSP430.

Need statement

- The purpose for this project was to create a programmable interface for controlling an industrial oven, refrigerator, or freezer.
- Industrial applications often require a set of steps where, for example, the temperature rises from room temperature to a given temperature, remains at that temperature for a given time interval, rises to another temperature, remains at that temperature for another time interval, and ultimately powers down.
- Microcontrollers with preloaded software already exist for these applications, but run in the range of several hundred dollars.
- Using the MSP430 with custom built temperature controller software greatly reduces the cost, and can be used again and again by the Engineering Design, Manufacturing, and Management Systems (EDMMS) department at WMU.

WESTERN MICHIGAN UNIVERSITY

College of Engineering and Applied Sciences

CLI

- 13 commands
- Based on requirements
- Handled over UART peripheral
- Executed using libemb shell library
- CLI input is parsed and validated
  - Only accepted parameters are positive integers
- Continuous loop until user exits the program

Command	Description
help	List available commands
PID a b c	Set 3 values for PID, respectively. P = a, I = b, D = c.
args	Prints back given arguments
+	Run the currently selected program from the currently selected step.
|	Pause execution and maintain the current setpoint.
-	Halt execution and turn the output pin off

CLI

Command	Description
p N	Load program N, step 0.
s N	Load step N of the current program.
. N	Set the target setpoint of the current step to N kelvins (only allowed when execution is halted).
m N	Set the interval of the current step in N minutes (only allowed when execution is halted).
v [01]	Reset/set verbose mode.
r N	Set reporting interval to N seconds.
show	Display the contents of the PID object.

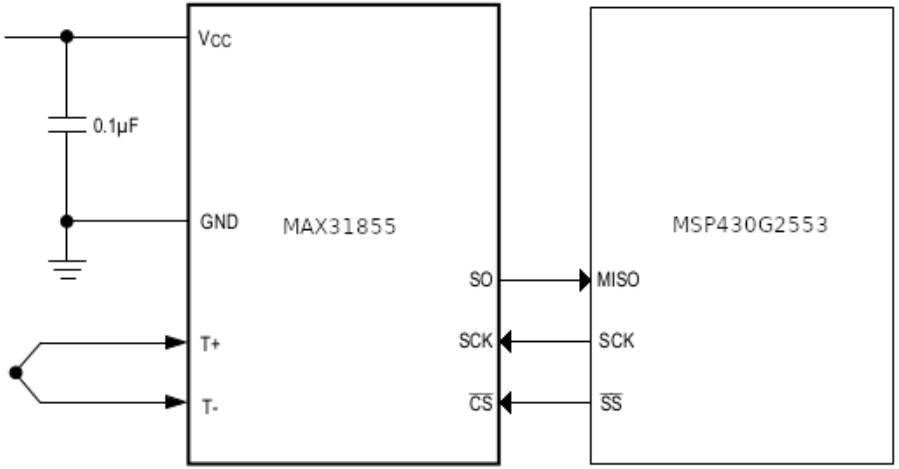
Data structures

- struct _PID_t:
  - Holds information gathered from CLI
  - Used for PID algorithm and temperature programs
  - Holds 9 member definitions
    - All of data type int
  - 36 bytes
    - MSP430G2553 only holds 512 bytes of data

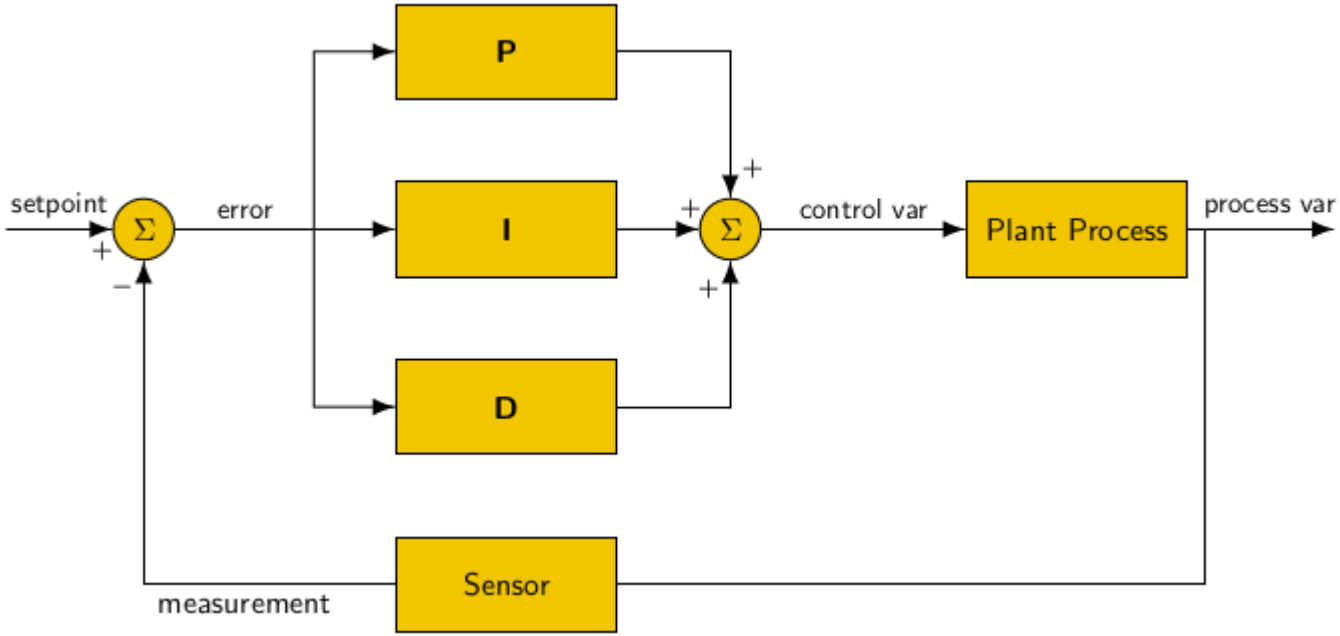
```
typedef struct _PID_t {
    int p;
    int i;
    int d;
    int v;
    int program;
    int step;
    int setpoint;
    int stepinterval;
    int reportinterval;
} PID_t;
```

SPI

- Synchronous serial communication interface used to communicate with the MAX31855 from the MSP430.
  - MSP430 sends clock signal from SCK to SCK on the MAX31855 and clears SS to select it.
  - MAX31855 sends 32-bit temperature data from SO to MISO on the MSP430.
    - We read the top 14 bits to get the temperature data



PID algorithm



PID algorithm

PID continuous equation, parallel form

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

PID difference equation

$$u[n] := p[n] + i[n] + d[n],$$

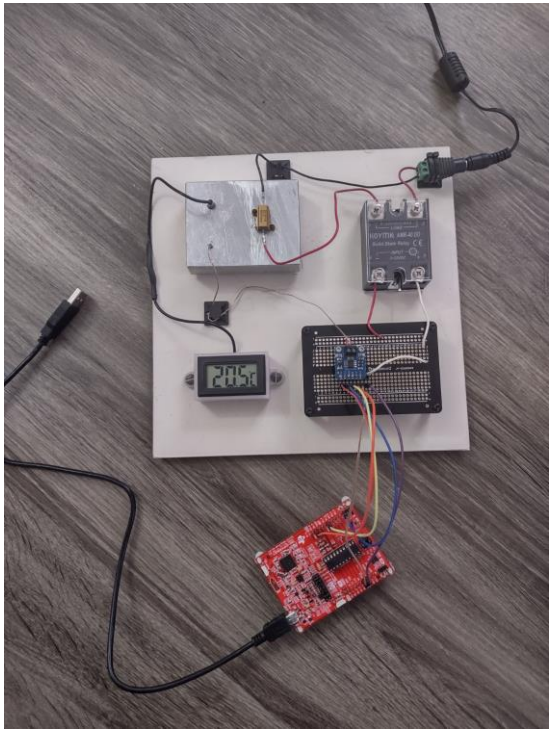
$$p[n] = K_p e[n]$$

$$i[n] = \frac{K_i T}{2}\left(e[n] + e[n-1]\right) + i[n-1]$$

$$d[n] = \frac{2K_d}{2\tau + T}\left(e[n] - e[n-1]\right) + \frac{2\tau - T}{2\tau + T} d[n-1]$$

Test Beds

- Working off campus due to Covid-19, we were unable to use industrial ovens or freezers for development and testing
- For this reason, our client developed the test bed to the right
- It allows the MSP430 (red) to connect to our Thermocouple-to-Digital Converter (blue chip, bottom right)
- The thermocouple (thin copper-colored wire, center) provides a reading from the vicinity of the heating element (top left), which is powered by a Solid State Relay (top right)
- A digital thermometer (bottom left) is also attached to create a temperature reading to compare with the reading from our Thermocouple-to-Digital Converter



Problems Encountered

- Rusty with MSP430 development
- Experience with MSP430 development from CS 2230 Computer Organization and Assembly Language, but needed to be brought up to speed
- Inability to do any meaningful testing or development other than the shell without test beds
- It took our client a few months to order the parts, perform soldering and create the test bed prototype
- After creating the first test bed, our client realized that it could heat up to 400 degrees Fahrenheit
- The client had to create his own thermal paste to prevent a potential fire hazard
- When we received the remaining test beds, we had about a week to attempt to implement SPI communication, incorporate our PID algorithm, and test the whole process
- We ultimately implemented SPI communication, but were unable to incorporate the PID algorithm due to time constraints

Demo

- Our implementation uses a timer interrupt to display a single temperature reading from the test bed
- A continuous reading is not displayed as we had hoped
- We did not have time to implement all of the shell functions
- Instead, the temperature is displayed after calling any shell function

Conclusion

- Working shell with validation
- Shell updates controller data structure
- SPI communication implemented
- PID algorithm theory studied and documented
- Unable to incorporate and test PID algorithm due to time constraints
- **Future Recommendations:**
  - Incorporate PID algorithm
  - Connect data structure elements to required functionality
  - Create usage documentation

References

MSP430x2xx Family User's Guide (Rev. J):
https://www.ti.com/lit/ug/slau144j/slau144j.pdf

MAX31855 Cold-Junction Compensated Thermocouple-to-Digital Converter:
https://datasheets.maximintegrated.com/en/ds/MAX31855.pdf