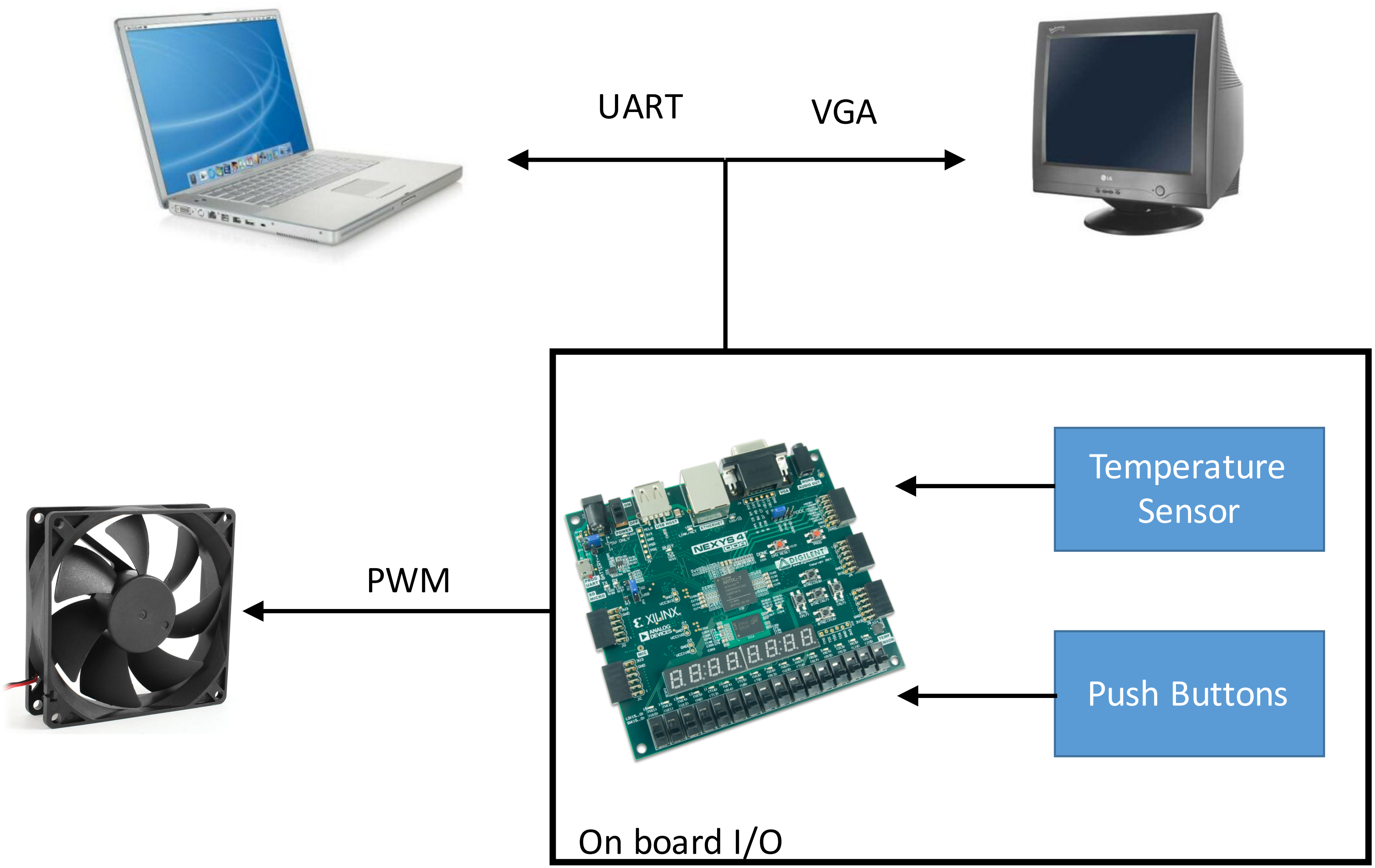
FPGA Temperature Regulation using PID Control in VHDL

David Paquette

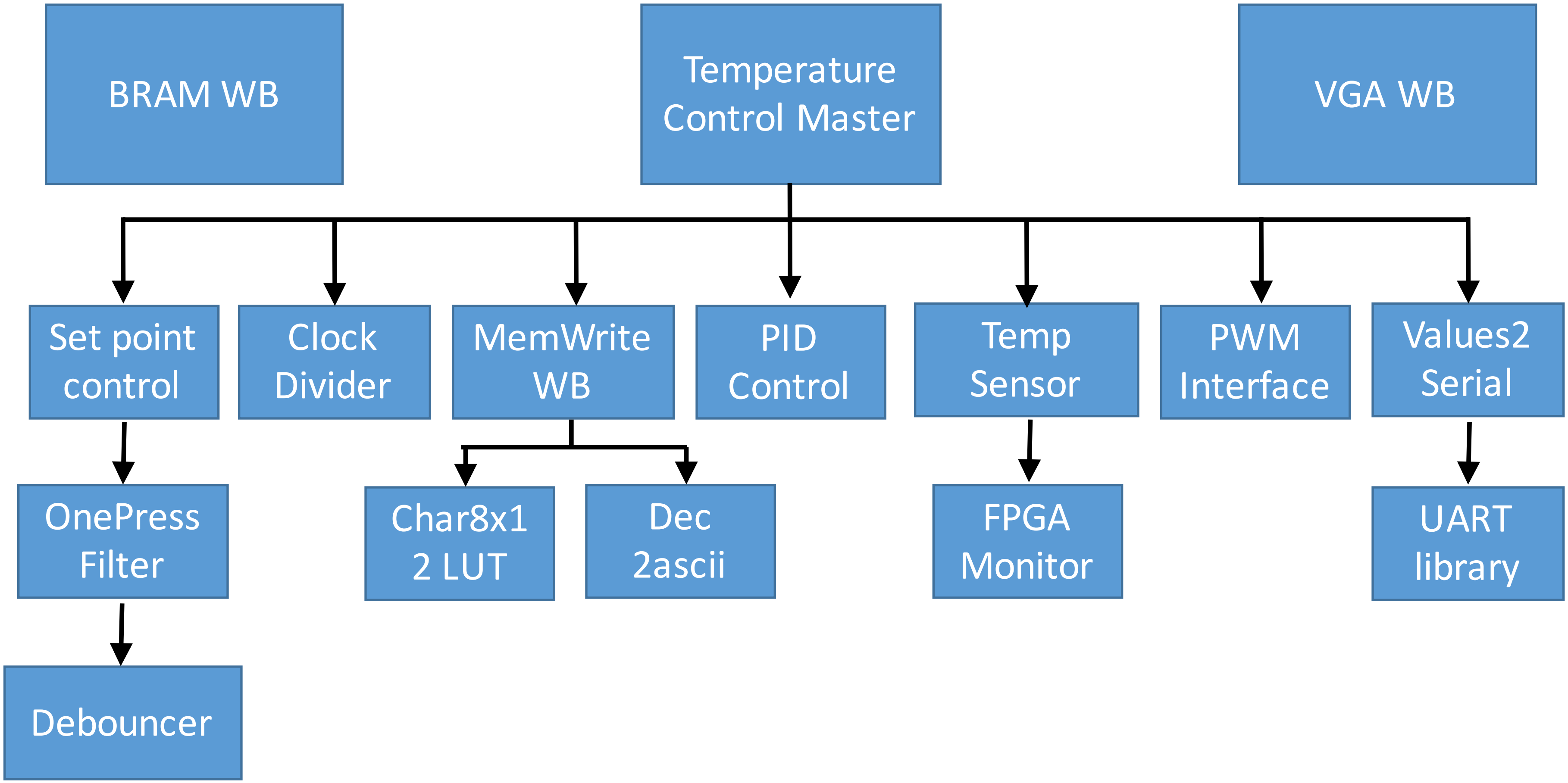
Goal

- Regulate the temperature of the FPGA using on board temperature sensor and external DC fan
- User selectable desired temperature
- View the current and desired temperature on an external display
- Collect current temperature and current fan speed on an external computer

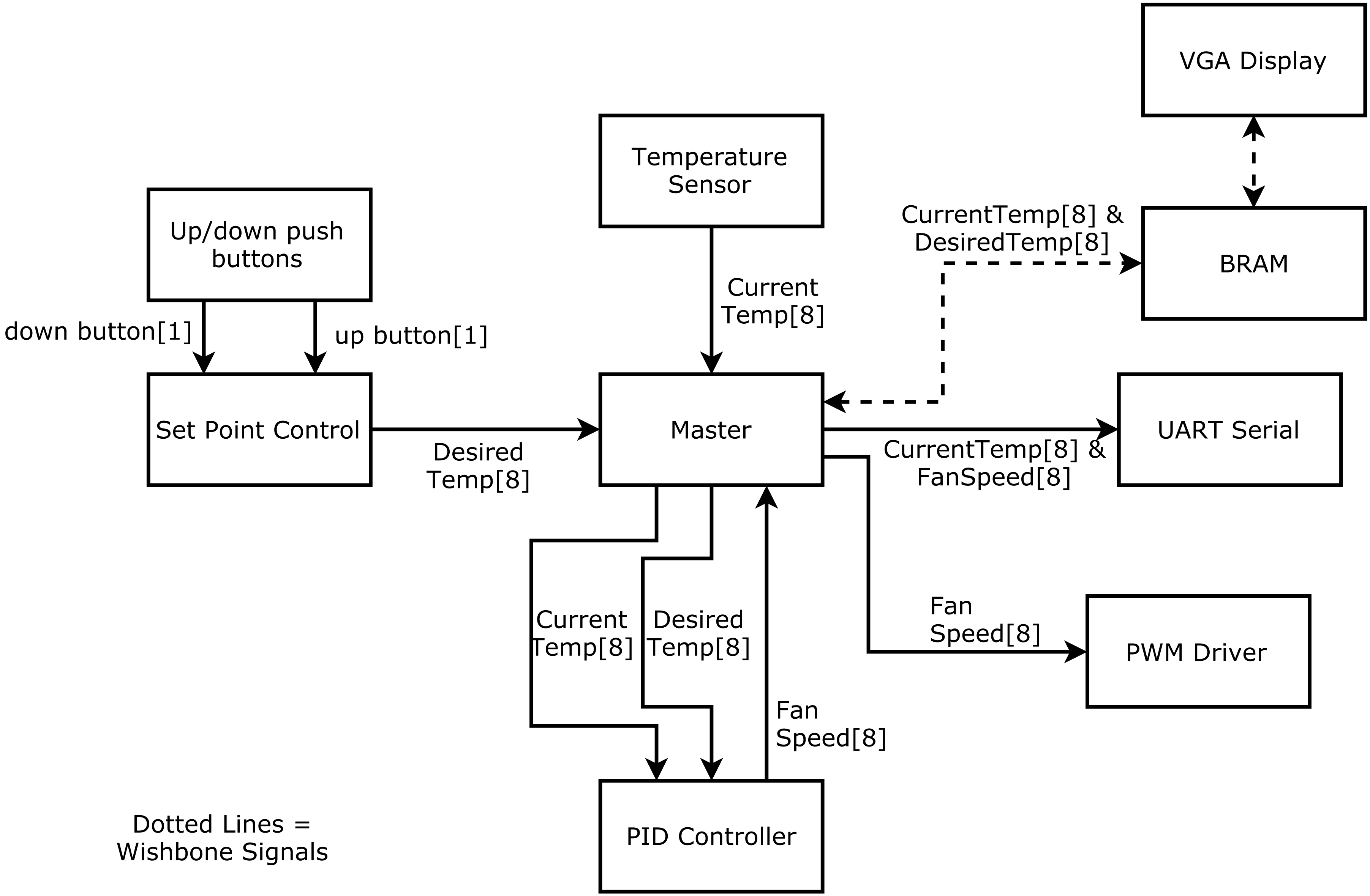
I/O Data Flow



High-Level VHDL Architecture

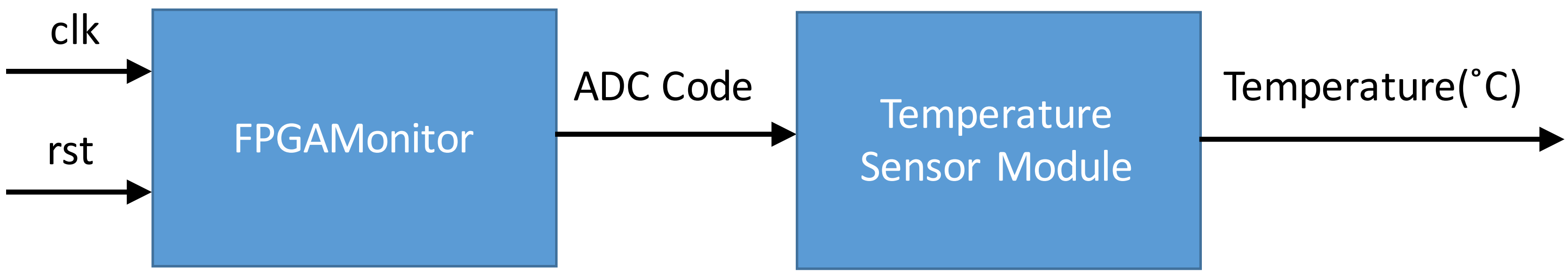


Internal Data Flow



Dotted Lines = Wishbone Signals

Temperature Sensor Module



Analog digital converter code to temperature

$$Temp(°C) = \frac{(ADCCode)503.975}{4096} - 273.15$$

Derived from $Voltage = 10\frac{kT}{q}ln(10)$

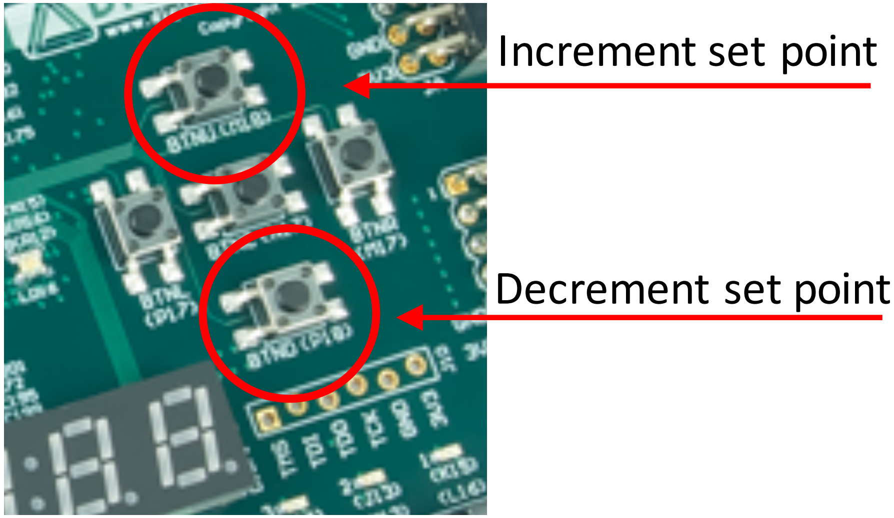
k=Boltzmann's constant
T = temperature (K)
q=charge on an electron

Voltage is sampled by the 12-bit ADC to produce an ADC Code

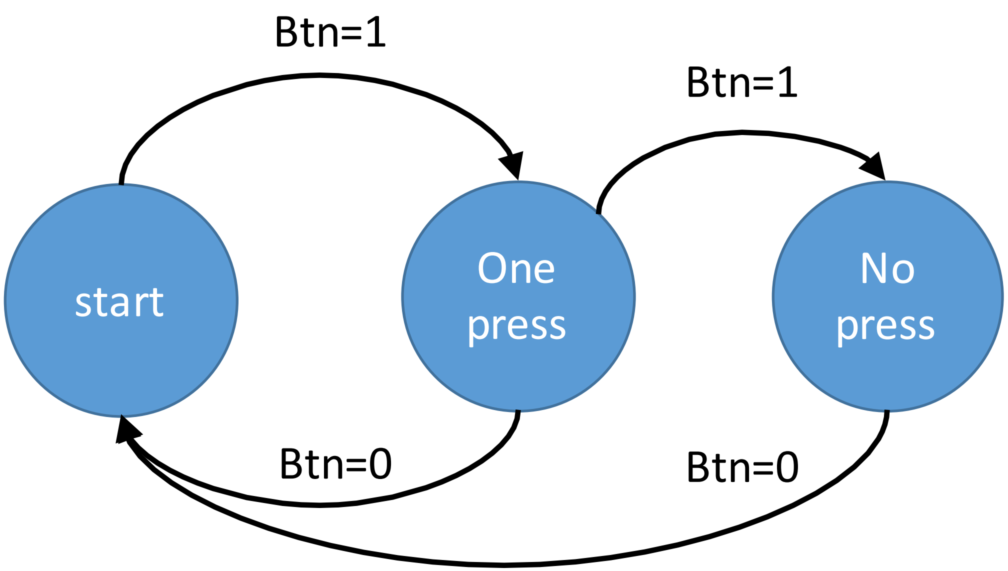
- Used XADC temperature sensor to measure the FPGA temperature
- For now, truncated temperature at 8 bits.
- Can measure in 1˚C increments

Equations found in Xilinx XADC user guide

Temperature Selection Module



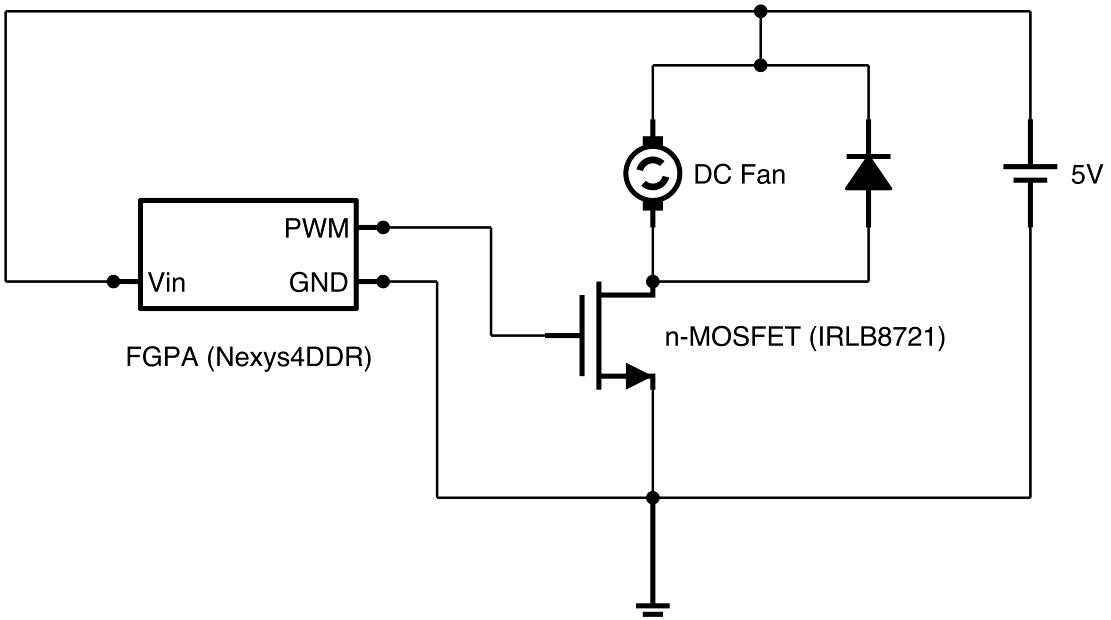
- Outputs the user's desired temperature
- Set by using the push button pad on the FPGA
- One press and debouncer state machine used



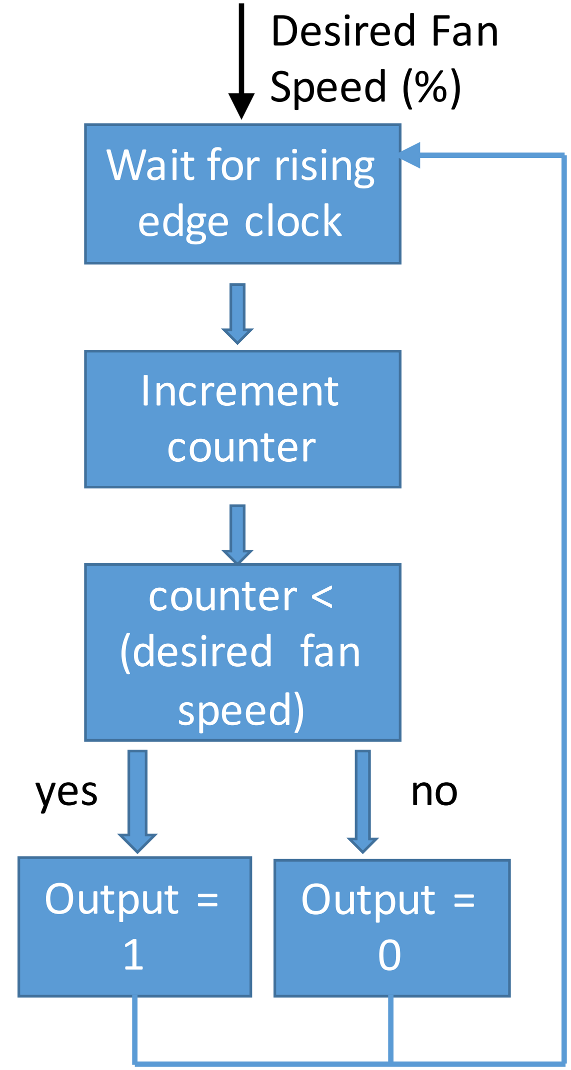
One press state machine

DC Motor Control Module

3.3V to 5V PWM DC Motor Driver



- Pmod digital output high is 3.3V
- DC motor is controlled with 0 to 5V
- DC motor functions in PWM frequency range 100Hz to 600Hz (from testing)



Duty Cycle % to PWM

UART Serial Module

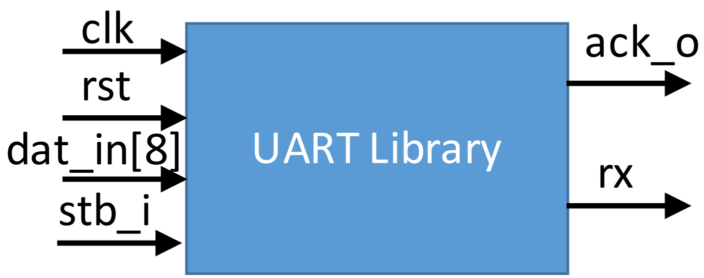
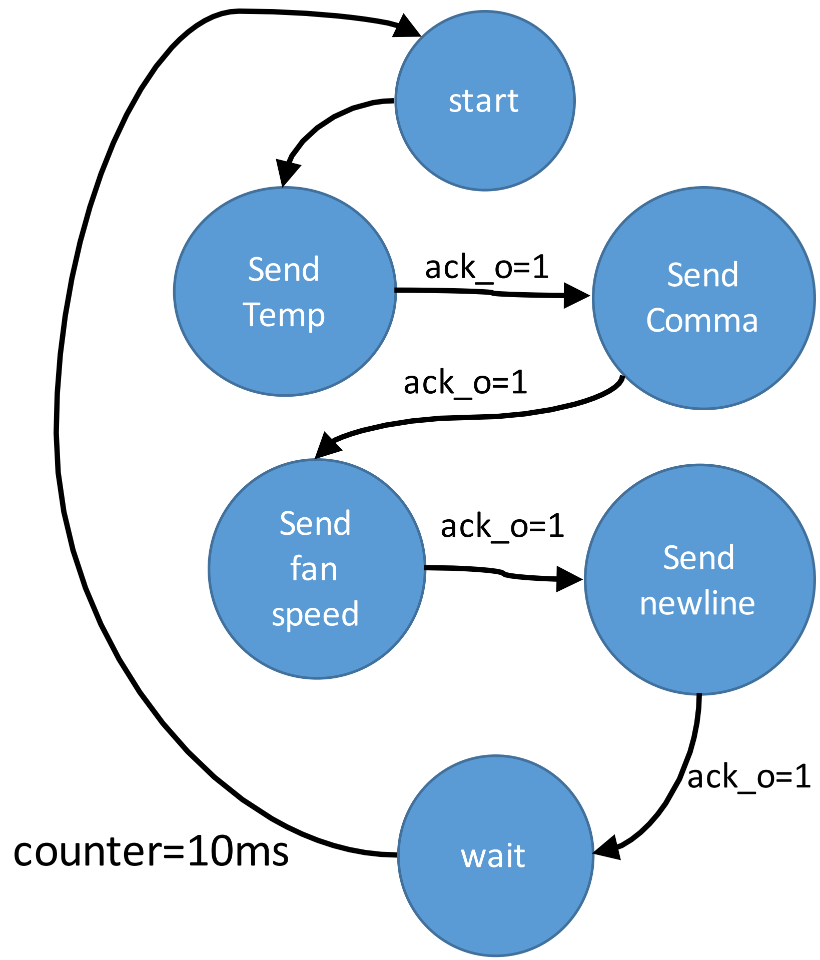


Diagram of UART library[1]
(only relevant ports shown)

Serial format used for sending data.
'X' is an 8-bit value (Current temperature and fan speed).
0x2C is ASCII for a comma (,)
0x0A is ASCII for a newline (\n)

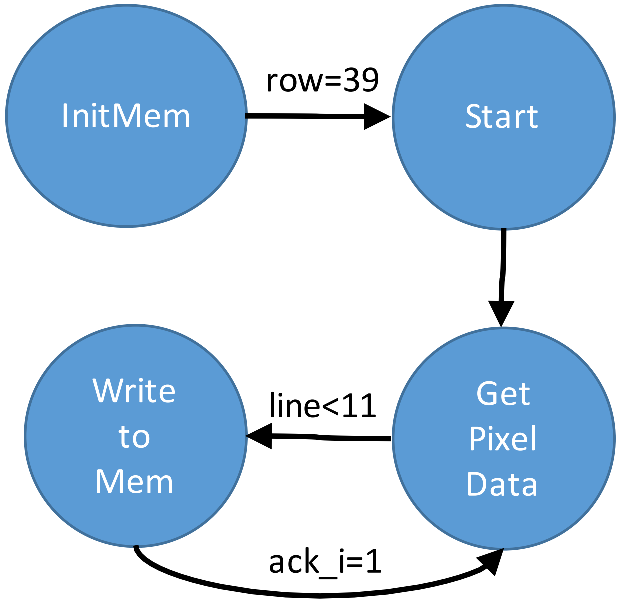


State machine for interfacing with UART library
Each state asserts the stb_i pin on entry and de-asserts it on exit.

1 http://bytebash.com/2011/10/rs232-uart-vhdl

VGA BRAM Display Module

- Current and desired temperature are written to BRAM over Wishbone
- VGA module reads pixel data from BRAM over Wishbone



BRAM Writer State Machine

$$ascii(tensDigit) = \frac{NUM - (NUM \bmod 10)}{10}$$

$$ascii(onesDigit) = NUM \bmod 10$$

Decimal to two digit ASCII conversion

cT=XX
dT=XX

Display format

Digital PID Controller Module

$$u_c[k] = K_p \cdot e[k] + K_i \cdot \Sigma\, e[k] \cdot T_s + K_d \cdot \frac{e[k]-e[k-1]}{T_s}$$

$e[k]=r[k]-y[k]$

$y[k]$ = measured temperature

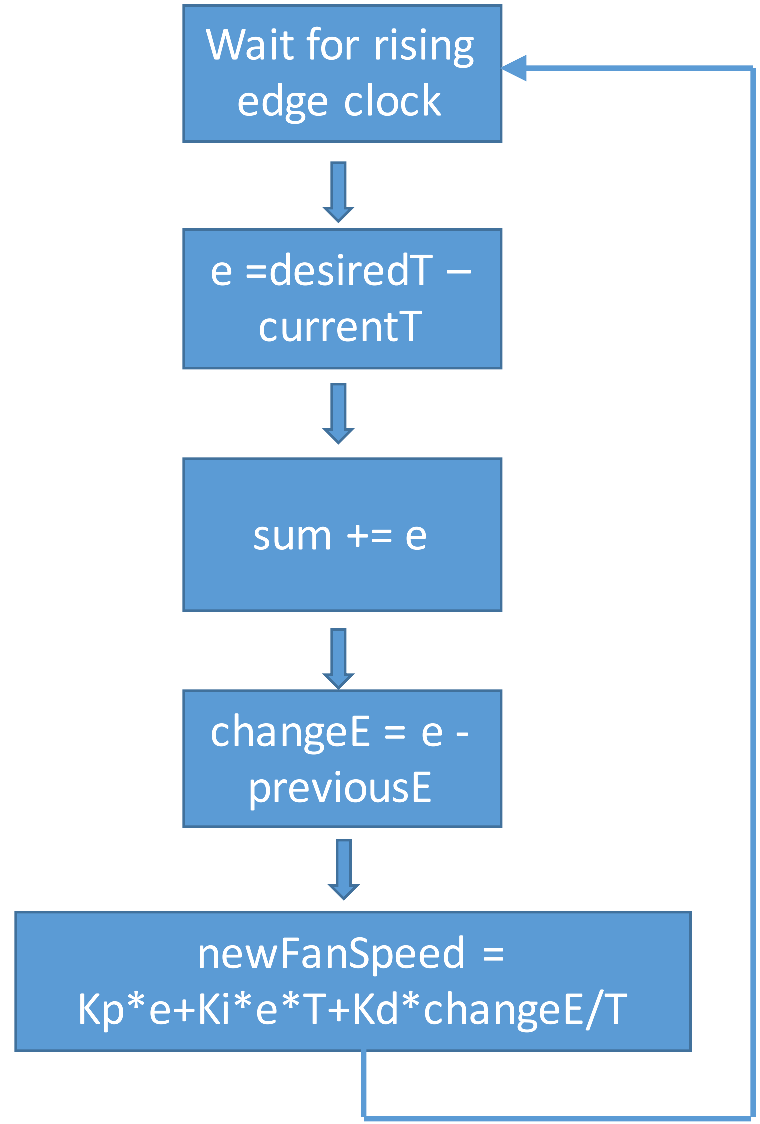
$r[k]$ = desired temperature

$u_c[k]$ = fan speed

T_s = 10ms

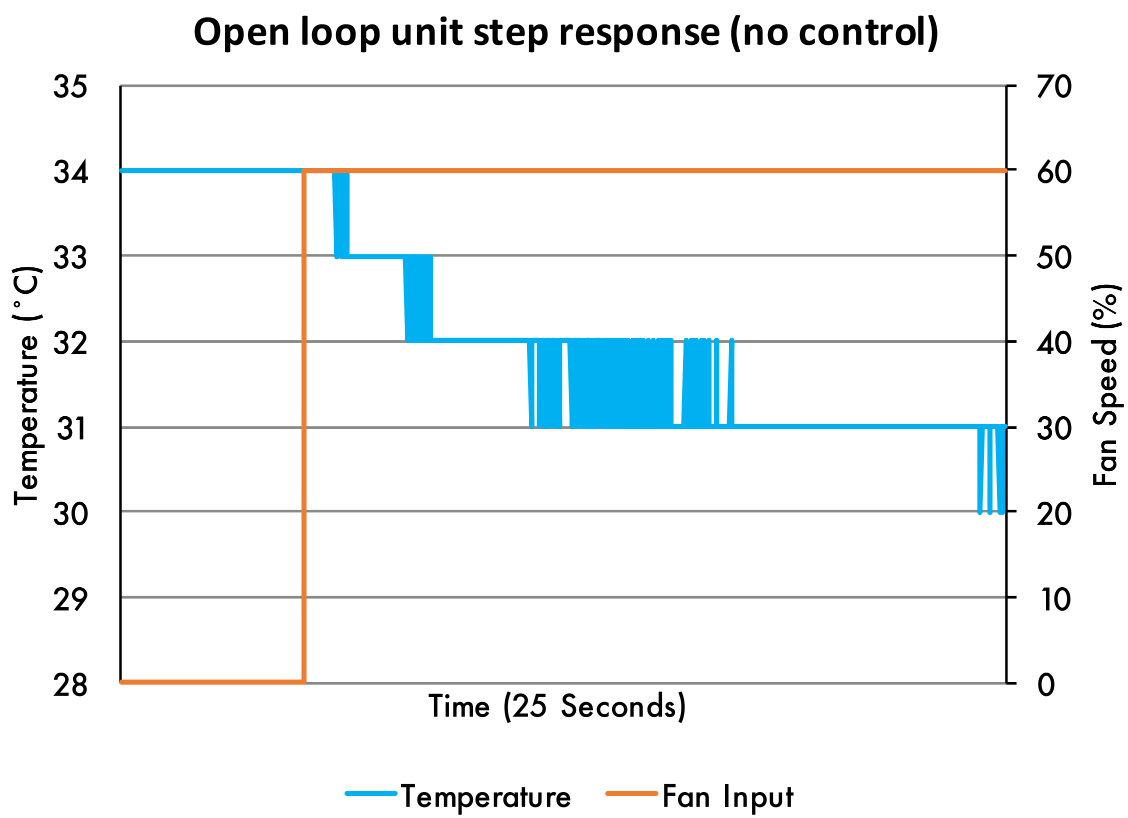
Safety Checks

- Added windup protection to prevent integral term from becoming too large
- Added output bounds checking to prevent fan speed from being set over 100% or under 0%



Simplified PID control

System Identification

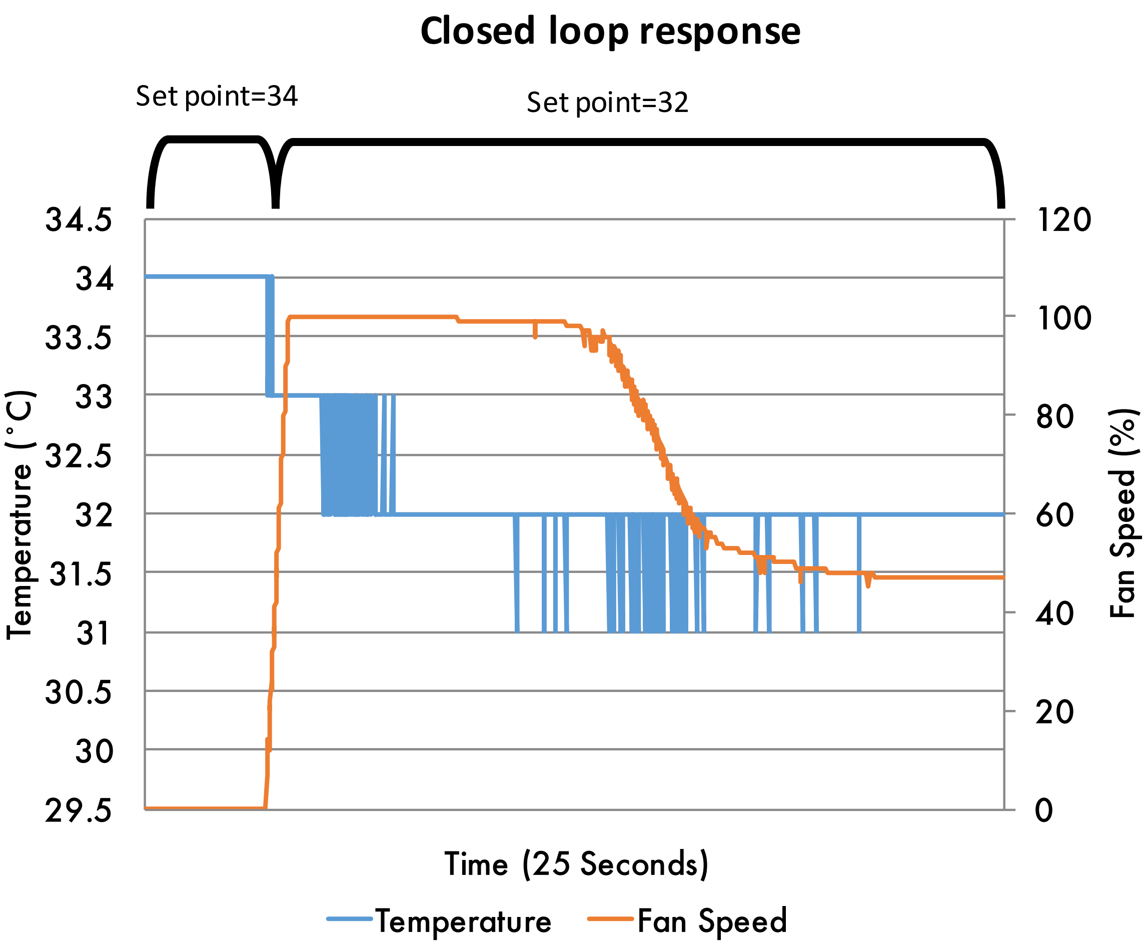


$$G(z) = \frac{-0.01055z^{-1}}{1 - 0.9978z^{-}1}$$

Open loop transfer function

- Set fan to 60% after 5 seconds and
- Captured temperature data through serial port
- Used the system identification tool box in MATLAB to estimate the discrete open loop transfer function
- Sampling rate of 10ms

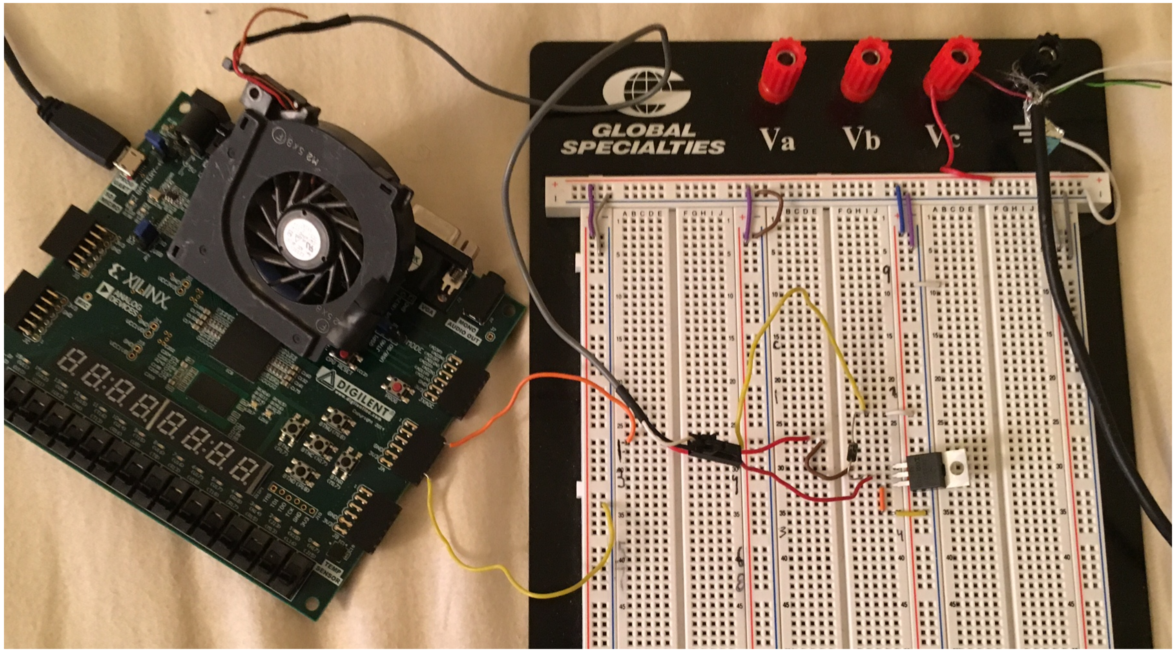
Controller Design and Results



$$K_p=-0.303 \quad K_i=-0.123 \quad K_d=0$$

- Used MATLAB's PID toolbox to estimate closed loop gains
- Tested gains on FPGA (worked well)
- Settling time of about 15 seconds
- To avoid using decimals, Kp and Ki are set as 303 and 123, then the controller output is divided by 100

Implemented Circuit and Display



Completed PWM DC motor driver circuit

VGA display format

Remaining Work

- Add module to generate heat, for testing controller response under varying conditions
- Limit desired temperature to reasonable range
- Display fan speed on VGA
- Move communication between fan, PID controller, temperature sensor, buttons and serial port to wishbone bus

Questions