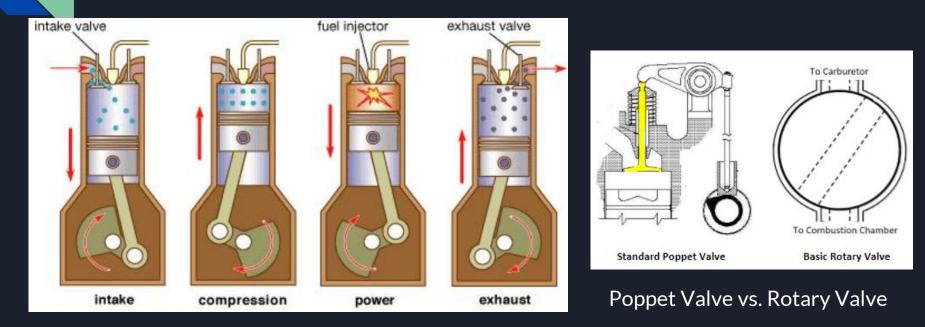
# Engine Control Unit

Chris Taylor, Robert McInerney

### (4 stroke) Internal Combustion Engine Basics



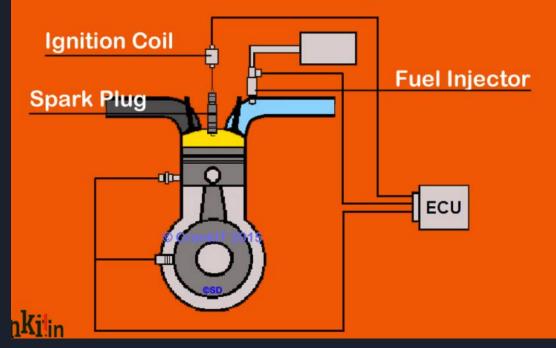
Combustion process requires precise timing

All events happen in relation to specific crank-shaft angular position

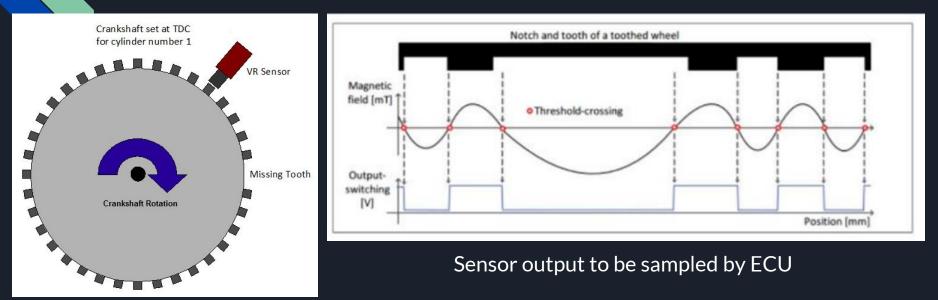


### Additional Peripherals

### **Electronic Fuel Injection**

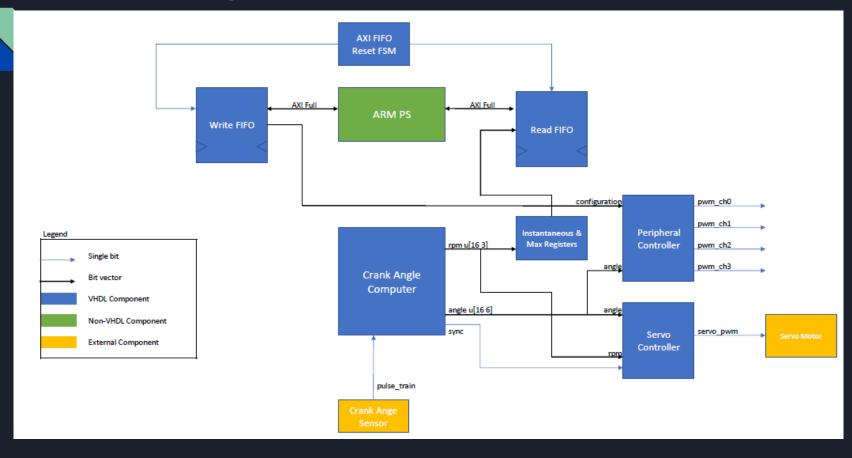


### Toothed Wheel and Sensor Design



60 tooth wheel, spinning at 5000 rpm = 100µs tooth width!

### Block Diagram



### **AXI Interface**

Writing:

- I/O channels are used to add new peripherals to the system quickly without diving into vhdl code
- Parameters given to the system:
  - Duty Cycle
  - Start Angle (NOT WORKING)
  - Stop Angle (NOT WORKING)
  - PWM Channel (0,1,2,3)
- Reading:
  - Current RPM
  - Max RPM reached

### Simulation - Crank Angle Computer (Sensor mimicry and synchronization)

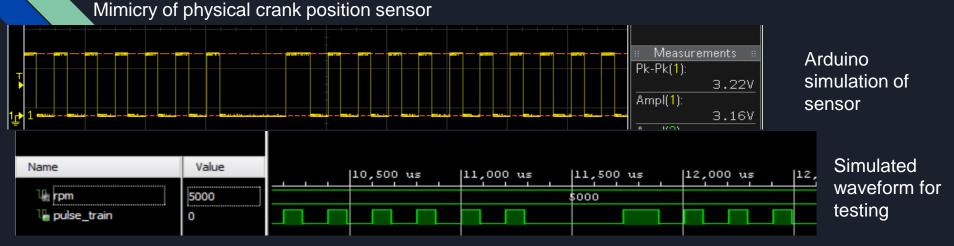
#### Mimicry of physical crank position sensor



#### Synchronization after 1 revolution

Name	Value	0 m.s .	10 ms	. 1	20 ms	30 m.s	40 ms .
E-Term test bench							
1∰a rpm	1000	5000		X		1000	)
Un rst	0						
₩ dk_125M	1						
🖬 📲 From Engine Simulator							
🔓 pulse_train	1						
From Crank Angle Computer	***************************************						
∎	42		۲	6789			
Un state	post_sync	pre_sync	X		po	st_sync	)
1 sync	1						

### Shortcomings - Implemented Hardware vs Simulation



System appears to be losing 'sync' resulting in pulse output of 0V.



### Simulation - Crank Angle Computer (Output)

Name	Value	12,000 us  12,200 u
tooth_count[5:0]	9	
La pulse_train	0	
U gap_present U tooth_count_rst	0 0	
₩ state ■ Magle[15:0]	post_sync 50.40625	
	4931.25	4816.5 X 4931.25 X



### Simulation - Peripheral Interface

Name	Value	0 m.s	10 ms .	20 ms .	30 ms .	40 ms
Verrpm Verrst	5000 0			5000		
₩ <mark>dk_125M</mark>	0					
CLK_125M_PERIOD	8000 ps 9a1a			000 ps		
	00fa 32	( (		00fa 32		
<pre>stop_angle[15:0] </pre>	8000 248a	0000		8000		
start_angle[15:0]     W pwm_out_ch1	0000 0	(		0000		
l∰ pwm_out_ch2 l∰ pwm_out_ch3	0 0					
U pwm_out_ch4	0					

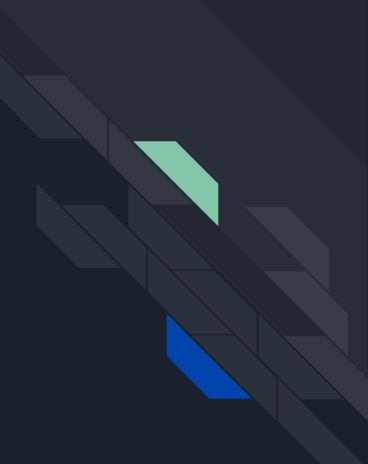
Pwm channels operate based off of start and stop angles of the system. The duty cycle can also be adjusted. This allows us to add new peripherals to the system at later date

### Simulation - Peripheral Interface

1 dock	0				
14 reset	0				
🕼 enab	0				
■ 📲 pwm_ch[2:0]	0		0		
🖬 📲 duty[7:0]	32		32		
🖬 📲 pulse[15:0]	04e2		04e2		
🖬 📲 angle_start[15:0]	0000		0000		
🖬 📲 angle_stop[15:0]	3423		3423		
current_angle[15:0]	0000			0000	
U pwm_out_ch1	0				
U pwm_out_ch2	0				
₩ pwm_out_ch3	0				
We pwm_out_ch4	0				
U CLK_125M_PERIOD	8000 ps		8000 ps		

We can see how the duty cycle changes based on input data

## Hardware Implementation



### Hardware Implementation - Peripheral Interface





#define	M1	0x327FFF20
#define	M2	0x005A021C
#define	MЗ	0x197FFF60
#define	M4	0x005A021C



### RPM Output via AXI

Instantaneous RPM:	5DCD
Max RPM reached:	5F2E 5DCD
Instantaneous RPM:	5000
Max RPM reached:	5F2E
Instantaneous RPM:	5DCD
Max RPM reached:	5F2E
Instantaneous RPM:	5DCD
Max RPM reached:	5F2E
Instantaneous RPM:	5DCD
<	

```
>> qf = quantizer('ufixed', [16 3]);
>> hex2num(qf, '5f2e')
ans
    =
   3.0458e+03
>> hex2num(qf, '5dcd')
ans =
   3.0016e+03
```

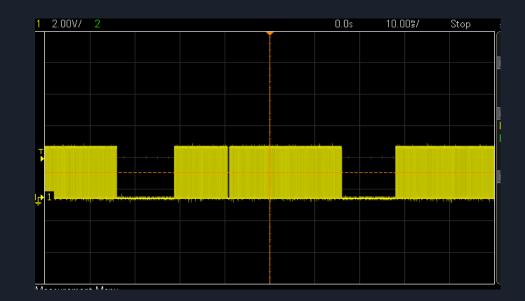
PS Console Output

Converted rpm via Matlab



### Challenges

- Sync Signal in HW
- Start/Stop Angle
- Accuracy of RPM
- Timing issues



Sporadic start/stop of PWM output due to losing 'sync'