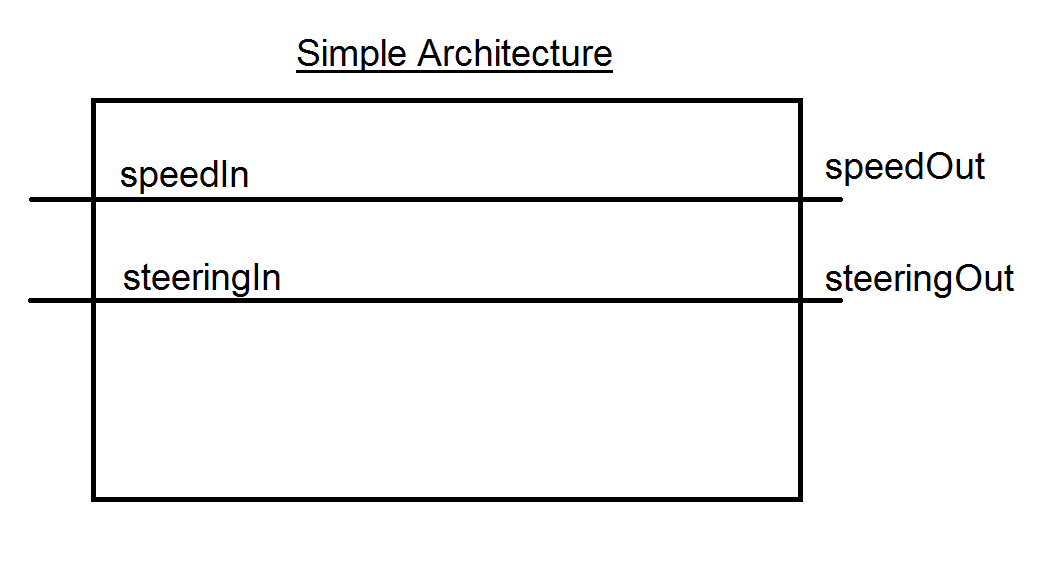
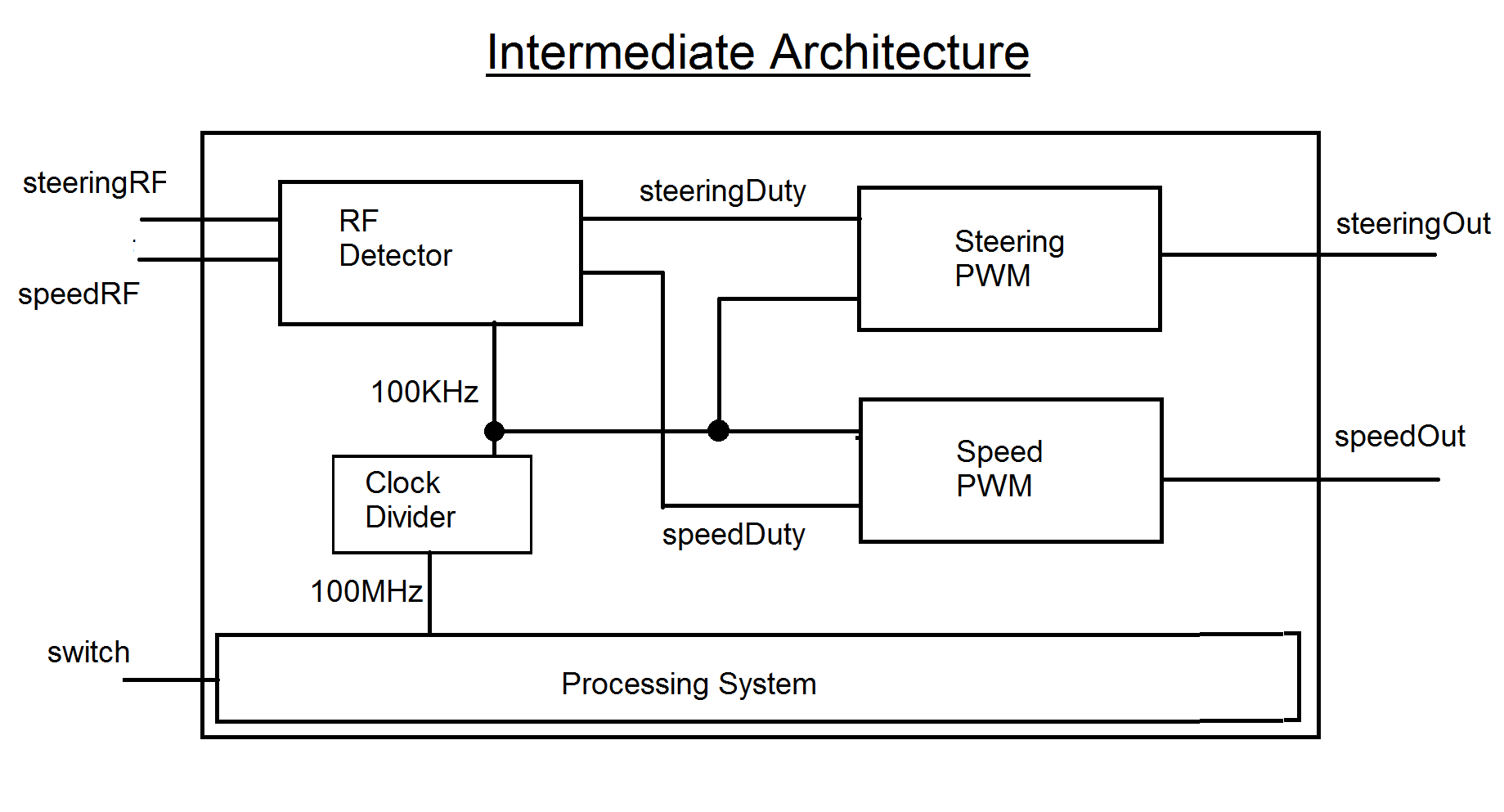
**RF route recorder**

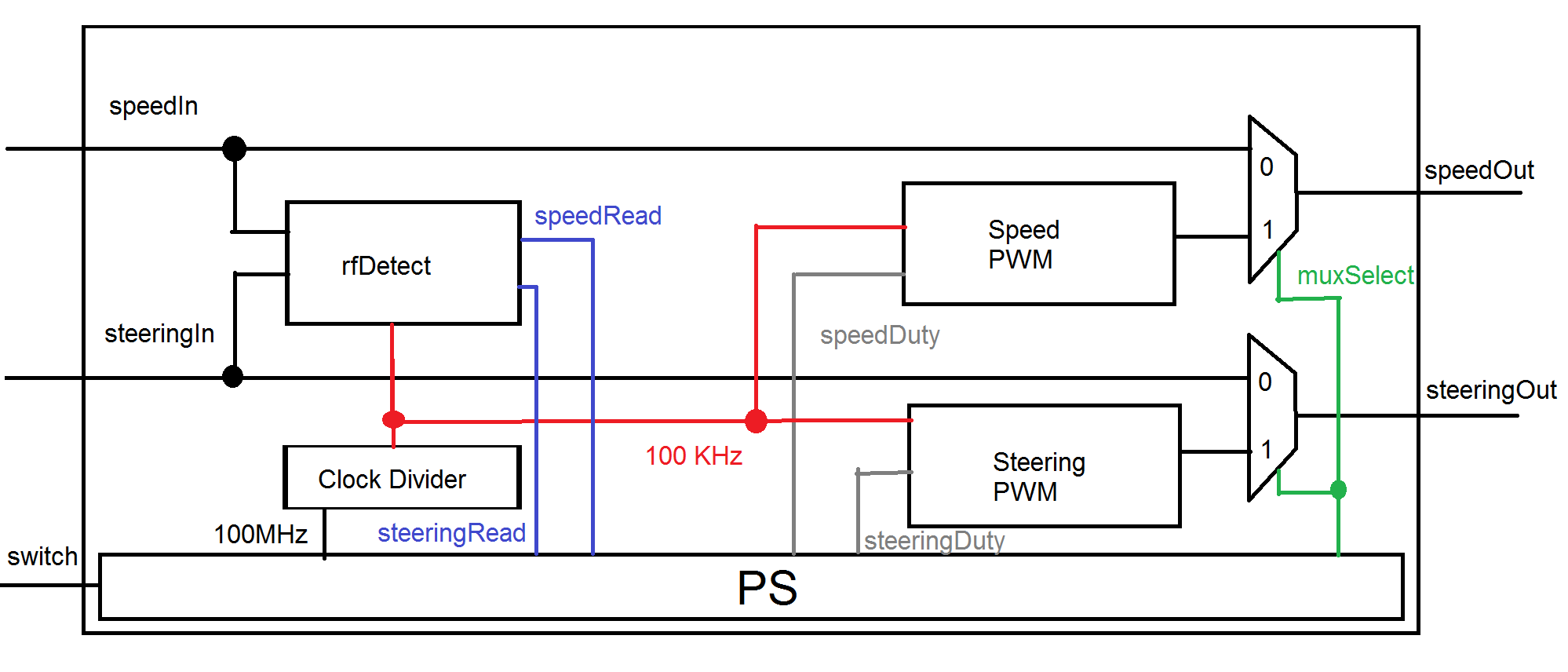
**Architectures:**



The simple architecture represents how to control the RazorCar without trying to save the data for replication, but simply takes in the PWM speed and steering signals and passes them directly to the actuators.



The intermediate architecture represents how to control the RazorCar without saving the data, but moves towards saving the data by adding components to read the speed and steering signals and convert them to an integer and two PWM components to convert an integer given to it into a PWM signal to control the actuators.



The advanced architecture represents how to control the RazorCar and save the data to be used to reproduce the given path. This architecture reads the steering and speed values and converts them to a vector to be processed by the processing system. The processing system then uses an input switch as an interrupt to determine whether to use the values from the remote or from memory to control the driving of the car. If the interrupt has not been triggered, the system will simply use the steering/speed values it is reading in from the remote. If the interrupt has been triggered, the control system will enter an infinite loop. If the switch is still high, the car will output a neutral duty cycle to the steering and speed PWM. If the switch is low, the car will read the duty cycles from memory to control the car.

**Components:**

1. RF Detector- Reads input from remote to determine the steering and speed duty cycle, outputs 2 vectors for the duty cycles to the PS
2. Clock Divider- Takes in a 100MHz clock and outputs a 100KHz clock
3. Steering PWM- Takes in a 100KHz clock and an integer and outputs a PWM output to control the steering
4. Speed PWM- Takes in a 100KHz clock and an integer and outputs a PWM output to control the speed
5. Processing System (xc7z020clg484-2)- reads in input from RF Detector and a switch to control the speed and steering of the RazorCar

**Tasks to complete:**

1. Setup – Due: 11/13/17

* Create project
* Create PS with proper AXI GPIO buses
* Create VHDL files for PWMs, Clock Divider, RF Detector, and Top level
* Begin researching how remote works to create a RF Detector that outputs the proper integer to the PWM to correlate with direction and strength of remote

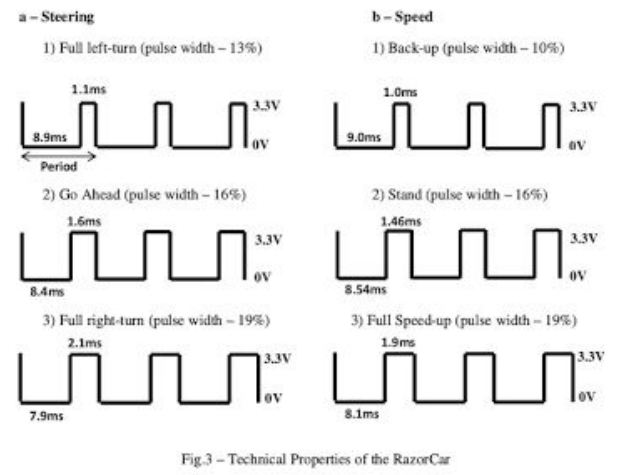
1. Drive car using remote – Due: 11/20/15

* Receive input from RF remote
* Process input to control car
* PWMs to control speed and steering
* Clock divider for PWM

1. Save path and replicate – Due: 11/27/15

* Research how to store values to memory for later use
* Modify architecture
* Store commands
* Signal to place car in standby mode while it is being moved
* Read stored commands
* Execute commands

**PWM:**



Steering: 110 – full left turn, 160 – straight ahead, 210 – full right turn

Speed: 100 – full speed reverse, 146 – neutral, 190 – full speed forward

**Pins:**

RC ­ Input NET

* “razorcar\_hardware\_0\_steeringIn\_pin" LOC = AA18 IOSTANDARD = LVCMOS33
* “razorcar\_hardware\_0\_speedIn\_pin" LOC = F17 IOSTANDARD = LVCMOS33

Actuators

* "razorcar\_hardware\_0\_steeringPwmOut\_pin" LOC = Y18 IOSTANDARD = LVCMOS33
* "razorcar\_hardware\_0\_speedPwmOut\_pin" LOC = T4 IOSTANDARD = LVCMOS33

SWITCH 1

* Switch\_pin <0> LOC = AA19 IOSTANDARD = LVCMOS33
* Switch\_pin <1> LOC = AB16 IOSTANDARD = LVCMOS33
* Switch\_pin <2> LOC = U11 IOSTANDARD = LVCMOS33
* Switch\_pin <3> LOC = AB21 IOSTANDARD = LVCMOS33