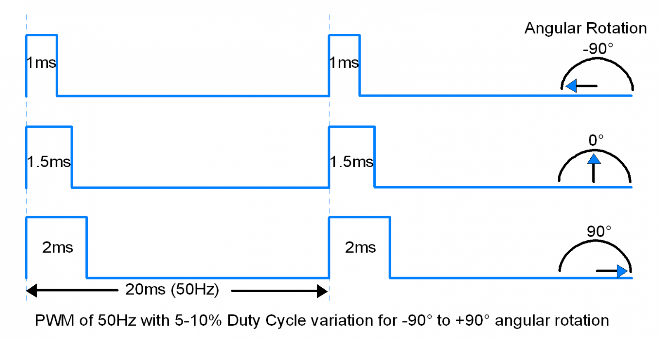
**Integration of Raspberry Pi with Servo Motor**

In order for the Raspberry Pi to interact with a deadbolt on a door the Pi needs a peripheral to mimic the motion of a hand turning the lock in two different directions, one for a locked position and another for unlocked. To achieve this, a high torque servo motor has been selected. The high torque allows for a stronger force to be applied into the turning motion that is sufficient enough to turn the latch. A servo motor was selected based on its capability to turn in different directions and the turning can be specifically adjusted to various angles. The servo motor connects to the Raspberry Pi pins via three different wires: power, ground, and control. Specifically, the servo requires a 5V power pin (pin 2), a ground pin (pin 9), and a general GPIO pin (pin 11) for the control signal. Due to the power consumption of the motor, it was elected that the power for the Pi be sourcing from a wall outlet rather than a battery to ensure operation and thus eliminating some of the power and energy constraints.

**Programming the Servo Motor**

The programming of the servo is not overly difficult; however, it is complicated by the fact that the servo is driven by **pulse width modulation** (pwm). The servo requires that it receives a control signal of 50 Hz. The position of the servo is determined by how long the signal is in the HIGH part of the cycle. However, when programming the Raspberry Pi, it is required that the **duty cycle** is given, which needs to be calculated. The complete cycle refers to how long the signal is high plus how long it is low. Consider the period of the signal, the period is equal to the complete cycle. If there is a duty cycle of 50% that means it is high 50% of the time. If it is 25% then this means it is high 25% of the time and low the other 75%. Recall that the period is equal to 1/frequency (1/50Hz). This means that the period is equal to .02 seconds or 20 milliseconds. In other words the servo position is updated every 20 milliseconds with a pulse between 1 or 2 ms. This is considered to be 5 and 10 % of the duty cycle because 1ms/20ms equals a 5% duty cycle. In order for the servo to conduct a counterclockwise turn the pulse width needs to be less than 1.5ms. When it is larger than 1.5ms it turns clockwise. As previously stated, the python code for the Raspberry Pi will require the duty cycle. Thus, to turn left takes 1ms/20ms = 5% duty cycle and 2ms/20ms = 10% duty to go to the right. Through experimentation it was found that the optimal duty cycle of left and right was 2% and 12%, specifically for 90% to the left and right. Figure 1 summarizes this discussion.



**Figure 1: Duty Cycle corresponding to turning the servo**

<http://www.electronicwings.com/sensors-modules/servo-motor>

In the python code on the Raspberry Pi, the GPIO pin of the control signal is specified as an output pin and designated with a GPIO.PWM(GPIO pin #, frequency). The design for the smart lock is treated as an interrupt. The state of the lock is kept as a binary value of zero or one. When the android phone is within range, the application sends a signal via Bluetooth to change the state from the locked position to the unlocked position. In essence, the Pi is always searching for the MAC address of the connected devices. If the MAC address is recognized then it interrupts and calls a function that alters the state of the lock. Once it disappears from the range then it interrupts again to return to the previous state. Note, that this is adjusted and made more sensitive by incorporating signal strength along with MAC address in order to trigger the interrupt handler to change states.