To Mention:

* AVR watchdog
  + Tutorial
* Delay for Mega program start before monitoring
* Delay on ready pin to prevent toggling when no power

**Arduino Nano & The “Selfie” Servo Code**

The code on the Nano is very simple, with the single task of deploying a servo motor to a defined position when a signal is received from the Mega. The signal is just a digital input of HIGH triggered by the Mega. The Nano waits for this pin to pull HIGH, deploys the servo, then waits for the pin to pull LOW again. When it does, the servo returns to its retracted position. The values for servo retracted position (20) and servo deployed position (100) were determined through a little testing. These values create an arc of ~80 degrees, keeping the “selfie” photo out of the way when retracted and directly in front of the PiCamera when deployed.

***SERVOCODE***

**Arduino Uno & The Heartbeat Monitor Code**

The code running on the Arduino Uno heartbeat monitor is almost as simple as that running on the Arduino Nano servo controller. There are 3 digital wires running between the Mega and the Uno. The first is an input to the Uno, and the Uno waits for this pin to pull HIGH for 5 seconds before beginning the main program, and it serves as the ready signal from the main sketch on the Mega. I included a 5 second delay on this input to prevent it toggling true/false when no power is applied to the Mega or the Mega is resetting for any reason. The second pin is also a digital input, and it received the heartbeat signal from the Mega. The last pin is a digital output from the Uno and plugs directly into the reset pin of the Mega. Pulling this pin LOW momentarily triggers a reset of the Mega.

Once the main program begins, it remains in a while-loop as long as a timeout doesn’t occur. When a “heartbeat” (digital pin pulling HIGH) is received, the timer resets, keeping the program within the while-loop. If a heartbeat isn’t received for 10 seconds, the digital output pulls low, resetting the Arduino Mega.

The other interesting feature, albeit probably not of much use in this case, is the AVR watchdog I also included in this sketch. I can’t imagine a situation where the code running on the Uno would freeze, but if this was to occur, the AVR watchdog would automatically reset the Uno itself, ensuring that the heartbeat monitor is always active. The AVR watchdog is a little-known feature on the Arduino, but it included by default in the backend AVR utilities. To accomplish this, I used a really nice tutorial by Simon Tushev (<https://tushev.org/articles/arduino/5/arduino-and-watchdog-timer>). A few elements need to be included to make the watchdog work properly and there are some pre-defined constants that need to be utilized to set the watchdog timeout interval. On line #5, you can see the inclusion of the watchdog library itself (“#include <avr/wdt.h>”), lines #14-#20 contain the setup operations that need to run once and are called during setup, and at multiple locations you can see “wdt\_reset(),” which is called throughout the sketch to keep the watchdog timer alive and prevent accidental reset.

***HEARTBEATCODE***