|  |
| --- |
| QE #1 (4hrs) 10Dec18  From <http://brettbeauregard.com/blog/2011/04/improving-the-beginner%E2%80%99s-pid-reset-windup/>  Reading documentation for PID library sections:   1. Sample time – the sample time will be set by the library in order to give smoother responses 2. Derivative kick – makes approximation using #1 IOT significantly reduce derivative spikes from instantons changes in input 3. On-the-fly tuning changes – computes in a discrete way in order to allow the tuning parameters to be changed without throwing off the response. 4. Reset windup mitigation – Arduino PWM takes inputs from 0-255…each term is limited to that range. 5. On/off (auto/manual) – PID control is either auto or off 6. Initialization – turning the PID control back on from manual…adds a method of saving last know state (Off|on) 7. Controller direction – set whether forward (^out=>^in) or reverse (opposite of forward)…ours is **forward**   Proportional on measurement – looks at the input for Kp and eliminates overshoot (prop on err requires –err to adjust => overshoot) |
| QE #2 (1hr) 10Dec18  Began code for Arduino: |
| QE #3 (1hrs) 11Dec18  From <https://www.arduino.cc/en/Reference/SD> (unless otherwise stated)  Reading documentation on SD library (SD = Secure Digital memory card)   1. Built on from “sdfatlib” created by William Greiman. 2. Supports FAT16 and FAT32 (File Allocation Table)    1. FAT16 can make partitions up to 4 GB ([Source](https://searchwindowsserver.techtarget.com/answer/Whats-the-difference-between-FAT32-FAT16-and-NTFS) same for a-d)    2. FAT32 can handle larger partitions and longer file names.    3. Neither offer built-in file security.    4. Neither offer good error recovery.    5. However, due to simplicity almost all systems can use FAT ([Source](http://forum.arduino.cc/index.php?topic=205949.0)) 3. Uses short 8.3 names for files    1. File names passed to the function can include paths separated by “/”    2. Version 1.0 supports opening multiple files. 4. Communication uses [SPI](https://www.arduino.cc/en/Reference/SPI) (Serial Peripheral Interface)    1. SPI on pins 50|51|52 for mega    2. SS (Slave Select) on pin 53 for mega (or specified using SD.begin() )       1. SS needs to be specified as pin or as output for function to work |
| QE #4 (1hrs) 11Dec18  From <https://learn.adafruit.com/adafruit-micro-sd-breakout-board-card-tutorial/arduino-library> (unless otherwise stated)  Reading on the implementation of SD library from Adafruit   1. File>Examples>SD in the Arduino IDE will get different examples of the functions of the library    1. CardInfo – reads info ABOUT THE CARD, can tell if the card is supported.    2. DataLogger – log data from three analog sensors ([Source](https://www.arduino.cc/en/Reference/SD) , same for a-f)    3. DumpFile – Read file    4. Files – Create and destroy a file    5. Listfiles – Prints out the files in a directory on a SD card    6. ReadWrite – Read/write data to/from 2. Once card is connected and sketch uploaded, run and open serial window    1. Check that “Volume type is FAT16” and the card size are correct    2. The library has prewritten comments about card insertion status    3. **Must “close()” each file when done to ensure permanence!**    4. Can open specific files to save into 3. Leaving a file open will take up RAM 4. [Other useful functions](https://learn.adafruit.com/adafruit-micro-sd-breakout-board-card-tutorial/arduino-library-docs)    1. “seek()” will move the r/w pointer to a new location => maybe use a UWP app to set seek(0)…it’s not deleting anything, just saying start from the beginning (i.e. write over the old)…could cause confusion if the old data set isn’t completely written over, though. |
| QE #5 (1hrs) 12Dec18  From <https://www.arduino.cc/en/Reference/SPI>  Reading on SPI library from Arduino   1. Serial Peripheral Interface    1. For short distances 2. Lines    1. MISO – Master In Slave Out – Slave line for sending to Master    2. MOSI –Master line for sending to peripherals    3. SCK – Serial Clock – Clock pulse for synchronization    4. SS – Slave Select – Used to (dis)enable each device ( 1 == EN) 3. Notes:    1. Arduino will automatically use the best rate (equal or less than what you set in SPISettings)    2. Can choose between MSB (Most Significant Bit) or LSB (least Sig. Bit) shifting…(second parameter in SPISettings)    4. SPI.beignTransaction(SPISettings(14000000, MSBFIRST, SPI\_MODE0))… SPI.endTransaction()       1. Stops all other SPI transactions until end       2. Could use begin, SS=0…SPI.transfer() (as much as needed)…SS=1, end… |
| QE #6 (1hrs) 12Dec18  From <https://en.wikipedia.org/wiki/8.3_filename> (the link Arduino listed to read)  Reading on 8.3-formatted character arrays   1. Used in old DOS and Windows (and for legacy compatibility) 2. Limited to FAT file system (with microprocessors) 3. Limited to 8 characters (not including any directory specifier) 4. Can *optionally* add a file name extension ‘.’ And then up to 3 characters (rest is ignored) 5. 8.3 is case-insensitive, but modern systems are not (so use case-sensitive to be compatible) 6. See website for directory table laws   <https://www.arduino.cc/en/Reference/SDCardNotes> Arduino’s notes about SD cards |
| QE #7 (1.5hrs) 12Dec18  From Microsoft Visual Studio’s Youtube channel (<https://www.youtube.com/watch?v=JvvbBj4s9kw>)  Video on how to create UWP app in Visual Studio 2017   1. Can run on anything that runs Windows 10 (or newer) 2. No emulator for Xbox or IOT (can test natively for the rest) 3. API – Application Program Interface…still have them (where it make sense)    1. E.g. “XboxLive Storage” isn’t really applicable to your desktop) 4. Application structure     2. Applications Class – Main       1. Entry point - what’s launched       2. Lifecycle management – when is it in use, when is it in the “background” for the user       3. App-wide resources – global stuffs (styles, etc.)       4. Unhandled exceptions - last road block to crashing    3. Window – “User-viewable window”       1. Minimize/close buttons go       2. Hosts frame (frame isn’t *required*, but is “very” common.)    4. Page – implements user interfacing (UI) and “app specific’ behavior       1. Two files: .xaml (UI definition) and .xaml.cs (associated code-behind file => behavior)    5. Manifest – app “metadata” for use by Store and Windows [XML]       1. Identity, visual assets (tile), capabilities (what we need), declarations. 5. Demo    1. **Created UWP basic project and will use this as the basis for creating the UWP app we need.**    2. ***I get to create the tile that will go on the start menu! That is very exciting.***    3. <https://docs.microsoft.com/en-us/windows/uwp/design/style/color> (how to choose theme of window) |
|  |
|  |
|  |