List of Basic Components Required for TVC:

* Microcontroller
* Inertial Measurement Unit (IMU)
* Barometer
* Data Recorder
* State Indicators
* Battery
* Parachute Stuff

Microcontrollers:

* Raspberry Pi
  + Pros:
    - High Processing Power (1-1.5ghz, 4-8 cores)
    - Built in Wi-Fi and Bluetooth for telemetry
    - Full Linux OS
    - Large community and support
  + Cons:
    - High power consumption
    - GPIO pins are controlled by the OS, which isn’t optimized for real time tasks, as control signals will have to pass through multiple layers of software before reaching the servos
    - Could be unpredictable timing in control loops, if OS decides to prioritize another process
    - It’s also kind of overkill processing power for what I need
* Beagle bone
  + Pros:
    - Real-time PRU (Programmable Real-Time Units) for precise control
    - High Processing Power (1 GHz)
    - Built in sensors like IMU, barometers, and motor drivers
    - Many GPIO pins and expansion options
    - Built-in battery management on some models
  + Cons:
    - More complex to program
    - Higher power draw
    - Bulkier than other microcontrollers
    - Fewer community resources
* Arduino Uno
  + Pros:
    - Very simple and beginner-friendly
    - Large community and wide library support
    - Low power consumption
    - Easy to program
    - Low cost
  + Cons:
    - Low processing speed (16MHz, 8 bit)
    - Limited RAM (2KB) and flash (32KB)
    - No built in IMU or wireless communication
    - Struggles with complex control algorithms
* Teensy 4.0/4.1
  + Pros:
    - Extremely high processing speed (60 MHZ, 32-bit)
    - Floating-point unit (FPU) for fast math calculations
    - Large RAM (1 MB) and flash (up to 8 MB)
    - High-resolution PWM
    - Compact size and low power consumption
    - Excellent library support
  + Cons:
    - Slightly more expensive than Arduino Boards
    - Fewer built-in peripherals (no Wi-Fi or Bluetooth)
    - Higher power draw than Arduino Uno/Nano
* Arduino Nano/Micro (Micro has more I/O pins)
  + Pros:
    - Very small and lightweight
    - Low power consumption
    - Inexpensive and widely available
    - Simple programming and wiring
  + Cons:
    - Low processing power (8-bit, 16 MHz)
    - Limited RAM (2KB) and flash (32 KB)
    - No built-in IMU

Overall Pick: Teensy 4.0

For it’s high processing power, high resolution PWM, compact size and lower power consumption. Fairly low priced, easier to program compared to some of the other boards, lots of resources online. I will need to integrate IMU sensors and telemetry modules as there isn’t built in Bluetooth or Wi-Fi.

Inertial Measurement Units:

* BNO055
  + Pros:
    - Integrated sensor algorithms
    - Easiest to use with basic microcontrollers
    - Includes magnetometer for absolute heading, reducing drift
    - Self-calibrating
    - 9 DOF
  + Cons:
    - Low update rate (100hz)
    - Higher latency due to on-board processing
    - Requires significant power
    - Less accurate due to internal processing limits
* BNO08x
  + Pros:
    - Advanced sensor algorithms with better drift correction
    - Higher output rates (1kHz)
    - Configurable algorithms for specific use cases
    - Better magnetic interference rejection
    - Lower power consumption than the BNO055
    - 9 DOF
  + Cons:
    - More complex setup and configuration
    - Requires more processing power
    - More expensive
* MPU 1650
  + Pros:
    - Super high sampling rates (32 kHz) and very low latency
    - Widely supported with good software libraries
    - DMP (Digital motion Processing) proesses raw gyroscope and accelerometer data to compute higher-level outputs like Euler angles or linear acceleration
  + Cons:
    - No absolute orientation without external magnetometer
    - More drift over time
    - More challenging to integrate without a dedicated fusion algorithm
* LSM6DS3
  + Pros:
    - Ultra-low power consumption
    - High output data rate (6.6 kHz)
    - Integrated finite state machine for custom algorithms
    - Small, lightweight
  + Cons:
    - No magnetometer (6 DOF)
    - More complex to implement advanced filtering and fusion
    - Higher data noise

Overall Pick: BN0O8x

I wanted to choose a 9DOF IMU which included a magnetometer to have better orientation accuracy and an absolute heading, which is critical for TVC control. This left me with either the BNO08x or the BNO055 as my options. However, the BNO055 has a data rate of only 100 Hz compared with the BNO08x of 1000Hz, which will be more suitable for real-time adjustments of the gimbal mount. In addition, this IMU includes integrated sensor fusion algorithms to provide orientation directly rather than requiring second-hand algorithms to derive the orientation. Although it is slightly more complex to set up and a slightly higher cost (manageable though), the overall flexibility and robustness of this IMU makes it the desired choice.

Barometers:

* MS5611
  + Pros:
    - Very high resolution (~10cm altitude resolution)
    - Low power consumption
    - SPI & I2C interfaces
  + Cons:
    - More expensive
    - Requires proper calibration and filtering for accurate results
* MPL3115A2
  + Pros:
    - Built-in altimeter function with altitude and temperature data
    - Mid resolution (30cm)
    - Low power consumption
    - Easy to integrate with microcontrollers
  + Cons:
    - Slow sampling rate (25hz)
* BMP388
  + Pros:
    - High resolution (8cm)
    - Fast data rate (200Hz)
    - Low power consumption
    - Built-in temperature compensation
  + Cons:
    - Requires careful calibration
    - More sensitive to temperature variations

Overall Pick: BMP388

Fast sampling and high resolution were the main reason why I chose this barometer. In addition to built-in temperature compensation functions and I2C and SPI interfaces, making it easy to integrate with the Teensy 4.0. The only cons is it can be sensitive to noise and requires careful calibration.

Types of Data Recorders:

* SD Card Modules
  + How it works:
    - Uses NAND Flash Memory where data is stored using floating gate transistors
    - Electrons are trapped to represent data, and stay trapped even when power is removed making it non-volatile
  + Pros:
    - High storage capacity compared to the others
    - Easy to obtain the data by simply removing the SD card
    - Widely supported across microcontrollers
  + Cons:
    - Physical interface and the mechanical connection of the SD card makes it more prone to flight vibrations
* Flash Memory Modules
  + How it works:
    - Very similar to SD card as it also uses NAND Flash Memory
    - Stores data using floating-gate transistors within a grid of memory cells
    - Cells act as switches controlling the flow of electricity to store binary data
    - Capable of storing multiple bits per cell allowing for high capacity in small form
    - It’s non-volatile meaning it doesn’t require power to store data
  + Pros:
    - High capacity
    - Fast read speeds
    - No moving parts making it resistant to vibrations
    - Simple integration with microcontrollers
  + Cons:
    - No standard file system (stores raw binary data)
    - More expensive than SD cards

Main difference between flash memory and SD cards is the mechanical vs. non-mechanical connection and built-in file system vs. raw binary data.

* FRAM (Ferroelectric RAM) Modules
  + How it works:
    - Data is written by applying an electric field across the ferroelectric capacitor
    - The direction of the polarization within the ferroelectric material represents the binary data
    - Main difference between FRAM and DRAM is DRAM stores data as an electrical charge in a capacitor.
  + Pros
    - Fast writing speeds
    - Non-volatile storage
    - not electromechanical making it resistant to vibrations
  + Cons:
    - Lower storage capacity

Overall Pick: Combination of FRAM module and SD Card module

Because the data on the SD Card is the most accessible out of all the options it makes the most sense to obtain the data from the SD Card. However, due to its mechanical connection, the data is sensitive to vibrations, which will inevitably occur during flight. The FRAM module being a non-volatile, solid-state device allows for high data recording rates with low noise. Compared with Flash memory, FRAM has near-instantaneous writes and is byte-addressable, so complex memory management isn’t needed. The one downside of using FRAM is the low storage capacity.

Calculate how much storage I would need

IMU Data: 6 axes (3D acceleration, 3D rotation) x 2 bytes per axis = 12 bytes

Barometer Data: 1 x 3 byte = 3 bytes

TVC Gimbal Angles: 2 angles x 2 bytes = 4 bytes

Gimbal Commands: 2 angles x 2 bytes = 4 bytes

Timestamp: 4 bytes

Flight Event Flags: 1 byte

Total: 28 bytes

Desired Sample Rate: 250hz

Bytes Per Second: 7000 bytes/s

Approximated Duration of Flight: 60 seconds

Total Storage Needed: 420,000 bytes

These are the modules we would use:

Adafruit SPI Non-Volatile FRAM Breakout - 4 Mbit / 512 KBytes - MB85RS4MT

Adafruit MicroSD breakout board PCB

Parachute Electronics: (will research more later, this is needed for sizing of the LiPo Battery)

* What do I need?
  + Something to command parachute deployment
  + a high-current switch to send a high current signal to the igniter
  + An e-match or nichrome wire to ignite the pyrotechnic charge
  + A redundant arming switch
* Microcontroller can be used to send signal to transistor
* For Switches:
  + Relay: electromechanical device that uses flowing current through a solenoid to create a magnetic field, which moves a mechanical switch. Allows us to switch from a low current signal from a microcontroller to high current for ignition of pyro charges
  + Relays in general are not a good idea to fly on rockets because the mechanical switch is prone to triggering due to vibrations
  + MOSFETS (Metal-Oxide-Semiconductor Field-Effect Transistor) is a type of transistor (used to amplify or switch electrical signals and power) that acts like an electronic switch.
  + They have very fast switching, high current capcity, and are solid-state so they are unaffected by vibrations of the rocket (much better option than the relays)
  + On a MOSFET there’s a gate that controls the flow of current through the MOSFET
    - The source is connected to ground and the drain is connected to the load (pyro igniter)
    - When the gate is activated a small voltage from the microcontroller creates an electric field which allows current to flow from the drain to the source

Telemetry:

* HiLetgo HC-12
  + Pros:
  + Cons:
* Adrafruit Feather M0 with RFM95
  + Pros:
  + Cons:
* Ebyte E32-433T30D

Summary:

Micrcontroller – Teensy 4.0

IMU – 2 x BN0O8x (one for recording gimbal angles, the other for rocket orientation)

Barometer - BMP388

Data Recorder - Adafruit SPI Non-Volatile FRAM Breakout - 4 Mbit / 512 KBytes - MB85RS4MT

+ Adafruit MicroSD breakout board PCB

Recovery – MOSFET + e-match or nichrome wire

Telemetry – 2 x Adafruit (PID 3072 RFM95W LoRa Radio Transceiver Breakout - 868 or 915 MHz (1 ground station, 1 on rocket)