

TF Electronics Throttle Controller

Software Installation:

Double click on TFEsetup.exe file to start installation. After installation there will be a shortcut on your desktop.

Connecting the USB cable for the first time:

Connect the USB cable to your PC and the throttle controller. The PC will recognize the controller and assign a port. Do NOT apply power to the controller.

From the toolbar select Start > Settings > Control Panel and double click on the System icon. Click the Hardware tab and then select Device Manager. Click on Ports and look for *CP210x USB to UART Bridge Controller*. Note the COM port number that this device is using.

Close all windows.

Software Setup Settings:

Double on the DBWsetup icon on your desktop to run the software.

Click Edit and then COM Settings from the menu. Select the com port number. Click Ok.

Apply power to the controller and press F3 to go online.

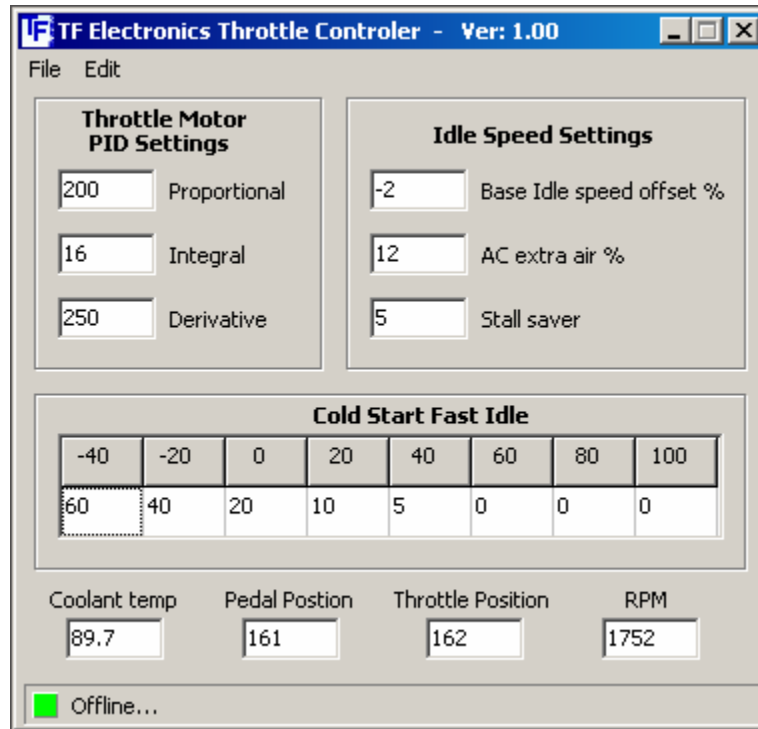
Click Edit and then TPS Calibrate from the menu. Follow the instructions to calibrate the throttle pedal range. When finished click close and then press F4 on your keyboard to lock the changes into the controller.

Your throttle controller is now ready to use.

Hotkeys:

F2.....Save calibration to a file
F3.....Go online or offline with the controller
F4.....Locks the changes into the controller

Software Settings:



Throttle Motor PID Settings:

See PID Tuning part of this manual for detailed information on PID's and how to tune these. It is recommended that you do not change these settings, as incorrect settings could cause the throttle control to be lost and cause excessive engine RPM.

Proportional: Default setting 200
Integral: Default setting 16
Derivative: Default setting 250

Idle Speed Settings:

Base Idle Speed Offset is the amount of throttle offset required to get the engine to idle at the RPM you require. This value is not in engine RPM.

AC extra air percentage is the amount of extra throttle opening when the Air Conditioner is switched on. This can also be used on car running antilag to open the throttle to get extra air.

Stall Saver sets the amount of extra throttle opening when the engine is decelerating. This setting times out after six seconds.

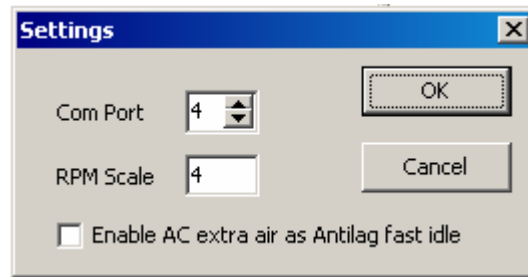
Cold Start Fast Idle:

The values in this table are added to the Base Idle Speed offset. These values are not in engine RPM.

Real Time Data:

These show the current coolant temperature and engine RPM. Also the throttle pedal and throttle butterfly position. These values are displayed in raw data from the controller. The value range is 0 to 1024, with 135 to 550 being normal values.

Setup Settings:



Com Port:

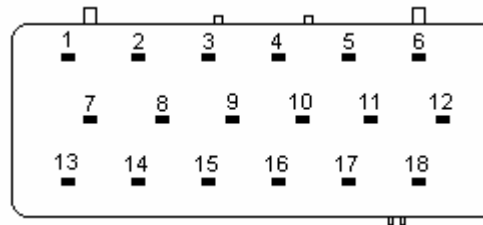
Sets the COM port assigned to the CP210x USB interface on the controller. See *Software Setup Settings* in this manual on how to set this.

RPM Scale:

Calibrates the RPM signal into the controller. For Subaru cars, 4 is the correct value.

Enable AC extra air as Antilag fast idle: Check this box to turn on this feature.

Pin-out:



Pin	Description	Subaru Pin
1	Throttle control motor (-)	B137-4
2	Throttle control motor (+)	B137-5
3	Sensor ground	B134-29
4	AC compressor on/off signal	B136-9
5	Auxiliary analog input	Not used
6	Throttle body Sub signal	B134-28
7	+12v Ignition switch I/P	B135-19
8	Ignition control relay output -ve	Not used
9		
10	RPM input signal	B136-22
11	Coolant temperature signal	B134-34
12	Foot pedal Sub signal (Diode required, see Fig 1)	B135-23
13	+12v Power supply	B136-1
14	Ground	B137-1 to B137-7
15		
16		
17	Foot pedal Main signal	B135-31
18	Throttle body Main signal	B134-18

ECU Throttle Position Signal:

The SM4 requires a throttle position signal. This signal is taken from the foot pedal sub signal. A signal diode is required to isolate the ECU TPS input from the Electronic Throttle controller input signals. Adding a 1N4148 small signal diode as shown below is required.

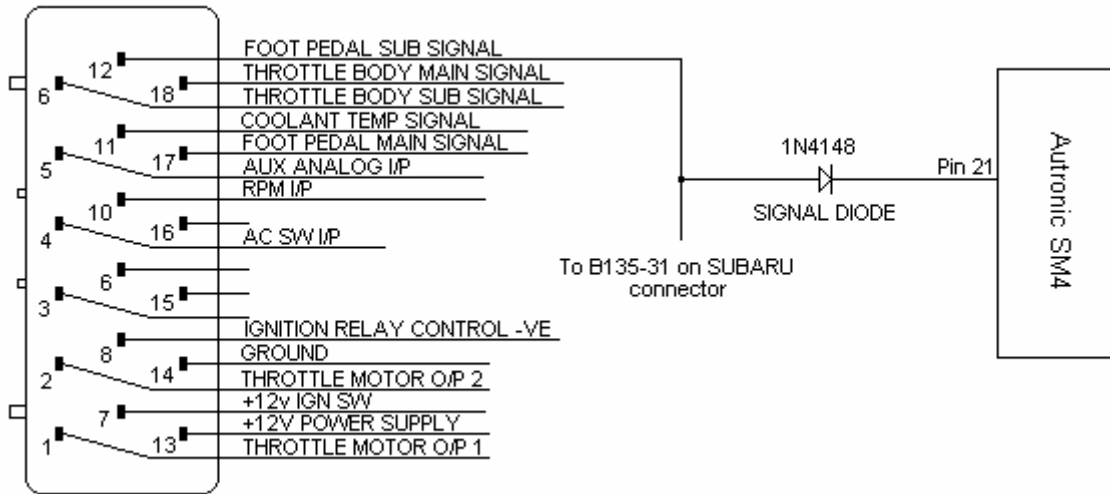


Fig 1

Understanding PID Control

Most control systems utilize feedback in some manner. Here's a look at several fundamental feedback mechanisms, culminating in a description of a basic PID controller.

Many real-time embedded systems make control decisions. These decisions are usually made by software and based on feedback from the hardware under its control (termed the "plant"). Such feedback commonly takes the form of an analog sensor that can be read via an A/D converter. A sample from the sensor may represent position, voltage, temperature, or any other appropriate parameter. Each sample provides the software with additional information upon which to base its control decisions.

Closed loop control:

Systems that utilize feedback are called closed-loop control systems. The feedback is used to make decisions about changes to the control signal that drives the plant. By contrast, an open-loop control system doesn't have or doesn't use feedback.

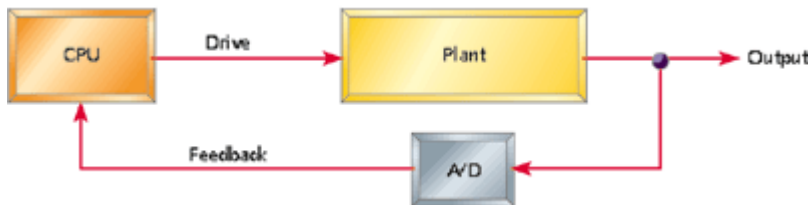
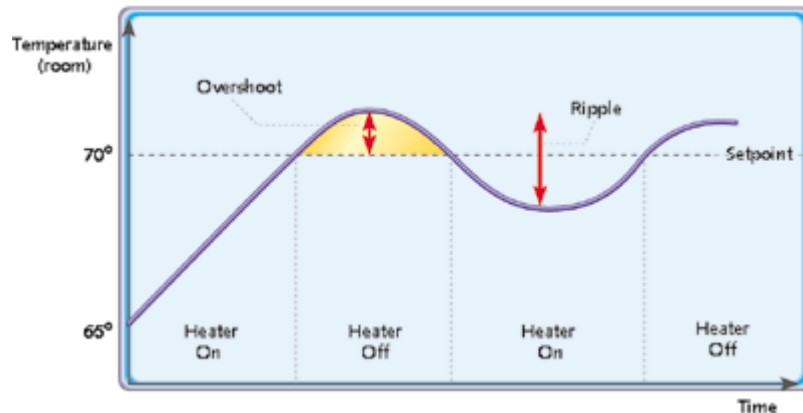


Fig 2

A basic closed-loop control system is shown in Figure 2. This figure can describe a variety of control systems, including those driving elevators, thermostats, and cruise control.

Closed-loop control systems typically operate at a fixed frequency. The frequency of changes to the drive signal is usually the same as the sampling rate, and certainly not any faster. After reading each new sample from the sensor, the software reacts to the plant's changed state by recalculating and adjusting the drive signal. The plant responds to this change, another sample is taken, and the cycle repeats. Eventually, the plant should reach the desired state and the software will cease making changes.

If feedback indicates that the temperature in your home is below your desired setpoint, the thermostat will turn the heater on until the room is at least that temperature. Similarly, if your car is going too quickly, the cruise control system can temporarily reduce the amount of fuel fed to the engine.



Bang bang

How much should the software increase or decrease the drive signal? One option is to just set the drive signal to its minimum value when you want the plant to decrease its activity and to its maximum value when you want the plant to increase its activity. This strategy is called on-off control, and it is how many thermostats work.

On-off control doesn't work well in all systems. If the thermostat waits until the desired temperature is achieved to turn off the heater, the temperature may overshoot. See Figure 2. The same amount of overshoot and ripple probably isn't acceptable in an elevator.

Proportional control is the primary alternative to on-off control. If the difference between the current plant output and its desired value (the current error) is large, the software should probably change the drive signal a lot. If the error is small, it should change it only a little. In other words, we always want a change like:

$$P * (\text{desired} - \text{current})$$

where P is a constant proportional gain set by the system's designer.

For example, if the drive signal uses PWM, it can take any value between 0% and 100% duty cycle. If the signal on the drive is 20% duty cycle and the error remaining at the output is small, we may just need to tweak it to 18% or 19% to achieve the desired output at the plant.

If the proportional gain is well chosen, the time the plant takes to reach a new setpoint will be as short as possible, with overshoot (or undershoot) and oscillation minimized.

Unfortunately, proportional control alone is not sufficient in all control applications. One or more of the requirements for response time, overshoot, and oscillation may be impossible to fulfill at any proportional gain setting.

A **Derivative** alternative

The biggest problem with proportional control alone is that you want to reach new desired outputs quickly and avoid overshoot and minimize ripple once you get there. Responding quickly suggests a high proportional gain; minimizing overshoot and oscillation suggests a small proportional gain. Achieving both at the same time may not be possible in all systems.

Fortunately, we do generally have (or can derive) information about the rate of change of the plant's output. If the output is changing rapidly, overshoot or undershoot may lie ahead. In that case, we can reduce the size of the change suggested by the proportional controller.

The rate of change of a signal is also known as its derivative. The derivative at the current time is simply the change in value from the previous sample to the current one. This implies that we should subtract a change of:

$$D * (\text{current} - \text{previous})$$

where D is a constant derivative gain. The only other thing we need to do is to save the previous sample in memory.

In practice, proportional-derivative (PD) controllers work well. The net effect is a slower response time with far less overshoot and ripple than a proportional controller alone.

Integration

A remaining problem is that PD control alone will not always settle exactly to the desired output. In fact, depending on the proportional gain, it's altogether possible that a PD controller will ultimately settle to an output value that is far from that desired.

The problem occurs if each individual error remains below the threshold for action by the proportional term. (Say the error is 3, P = 1/8, and integer math is used.) The derivative term won't help anything unless the output is changing. Something else needs to drive the plant toward the setpoint. That something is an integral term.

An integral is a sum over time, in this case the sum of all past errors in the plant output:

$$I * \sum_t (\text{desired} - \text{current})$$

Even though the integral gain factor, I, is typically small, a persistent error will eventually cause the sum to grow large and the integral term to force a change in the drive signal. In practice, the accumulated error is usually capped at some maximum and minimum values.

In summary, on-off and proportional control are the two basic techniques of closed-loop control. However, derivative and/or integral terms are sometimes added to proportional controllers to improve qualitative properties of a particular plant's response. When all three terms are used together, the acronym used to describe the controller is PID.