JOÃO PEDRO FREITAS DOS SANTOS
PEDRO DE CASTRO ROSSETTI

Otto III by FlexECU – Gerenciamento Eletrônico de um Motor VW 1.6L

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para a Conclusão do Curso de Engenharia.

Área de Concentração:
Engenharia Elétrica - Sistemas Eletrônicos

Orientador: Prof. Dr. Armando Antonio Maria Laganá

São Paulo
2015

JOÃO PEDRO FREITAS DOS SANTOS
PEDRO DE CASTRO ROSSETTI

Otto III by FlexECU – Gerenciamento Eletrônico de um Motor VW 1.6L

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para a Conclusão do Curso de Engenharia.

Área de Concentração:
Engenharia Elétrica - Sistemas Eletrônicos

Orientador: Prof. Dr. Armando Antonio Maria Laganá

São Paulo
2015

Agradecimentos

Ao estimado professor e orientador doutor Armando Antonio Maria Laganá, por todo o incansável esforço e dedicação empenhados neste grupo de Eletrônica Automotiva, fazendo o impossível para tornar possível o desenvolvimento desta área dentro do país e da universidade. Certamente será levado por nós como um exemplo de pessoa e profissional a ser seguido, alguém que não mediu esforços para que outros chegassem ao sucesso.

Também ao professor doutor João Francisco Justo que, junto ao professor Laganá, dedicaram parte de seu tempo à orientação à pesquisa desde o primeiro simples projeto 3 anos atrás até longas conversas sobre a vida acadêmica, profissional e conselhos sobre o futuro.

Aos queridos amigos da FATEC Santo André, que em parte se uniram a nós na Escola Politécnica recentemente, Bruno Silva Pereira, Demerson Moscardini, Paulo Hayashida, e demais colegas que sempre estiveram a postos a sanar dúvidas e colaborar conosco em todas as fases do projeto.

Ao querido amigo desenvolvedor do projeto Otto II, Bruno Fernandes, que serviu de base para este projeto, dando continuidade ao primeiro esforço realizado anos atrás com o projeto Otto pelos parceiros politécnicos André Soares e Vitor Scarpinetti.

Ao professor Francisco Salotti, por disponibilizar a estrutura do Instituto de Energia e Ambiente da USP, compreendida por um dinamômetro inercial de chassis, fundamental para todo o desenvolvimento e ensaios necessários.

À ETAS pela parceria com o grupo de Eletrônica Automotiva que tornou possível a realização deste projeto com a cessão de softwares e hardwares primordiais para este projeto. Esperamos que essa parceria continue evoluindo e dando frutos futuramente, pois só tem a contribuir com o desenvolvimento dos alunos e desta importante área em nosso país.

Por fim, a todos que contribuíram direta ou indiretamente com este projeto, desde um pequeno conselho até um apoio em momentos difíceis.

RESUMO

O presente projeto visa o desenvolvimento de um *software* de aplicação de uma ECU (unidade de gerenciamento eletrônico) para um motor Volkswagen 1.6L do ciclo Otto. Envolvendo todas as fases do controle do motor, este sistema foi inserido em uma FlexECU, mesmo hardware de uma ECU comercial de desenvolvimento.

Além do objetivo de aprimoramento dos conhecimentos, este projeto visa criar uma estrutura, através de uma implementação de uma arquitetura de software, que possibilite o desenvolvimento de novos projetos na área uma vez que um software aberto e modular permite total controle do sistema bem como desenvolvimento de partes especificas do controle.

Com o desenvolvimento desta estrutura, a equipe de eletrônica automotiva passa a contar com uma grande gama de ferramentas no estado da arte para o desenvolvimento, medição e calibração de *softwares* automotivos, podendo fomentar uma maior participação da indústria brasileira em todas as fases do desenvolvimento de softwares de aplicação frente ao atual cenário de grande atividade em medição e calibração e desenvolvimento de funções específicas para o mercado brasileiro como as que envolvem o sistema *flex* (bicombustível).

Palavras-Chave: Eletrônica Automotiva. Engenharia Automotiva. R&D, ECU, Ciclo Otto, V-Model, Model-based Software Development.

ABSTRACT

The following project aims the development of an application software of an ECU (Electronic Control Unit) for a Volkswagen 1.6L Otto Cycle Engine. Covering all the phases of the engine control, the system was inserted in a FlexECU, same hardware present in a commercial development ECU.

Besides the technical development objective, this project also aims to create a structure that facilitates new projects in the area as an open and modular software allows full control over the system as well as the development of specific parts of the control.

With the development of this structure, the automotive electronics team will have a big sort of state of the art tools for automotive software development, measure and calibration. It will allow a promotion of a bigger participation of the Brazilian industry in all parts of the application software development facing the actual scenario of major work on measurement and calibration as well as development of functions specific for the national market such as the flex system (dual fuel).

Keywords: Automotive Electronic. Automotive Engineering. R&D, ECU, Otto Cycle, V-Model, Model-based Software Development.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

8V	Oito Válvulas
A2L	Arquivo no formato *.a2l da norma ASAM MCD-2 MC
ADC	Analog to Digital Converter
AppSW	Application Software
ASAM	Association for Standardizationof Automation and Measuring Systems
ASCET	Advanced Simulation and Control Engineering Tool
BSW	Basic Software
CALC	Calculus
CAN	Controller Area Network
CCP	Can Calibration Protocol
cm	Centímetro(s)
cm^3	Centímetro(s) Cúbico(s)
CPU	Central Processing Unit
CTRL	Controller Area Network
cv	Cavalo(s)
ECU	Electronic Central Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
EHOOKS	Easy Hooks
EPM	Engine Position Management
EPUSP	Escola Politécnica da Universidade de São Paulo
ETB	Electronic Throttle Body
ETC	Electronic Throttle Controller
ETK	Emulator Tast Kopf
EV	Eficiência Volumétrica
g	Grama(s)
G5	Geração 5
GDI	Gasoline Direct Injection
HEX	Arquivo no formato *.hex que correspondente ao código compilado
HiL	Hardware-in-the-Loop
HW	Hardware
Hz	Hertz
IGN	Ignition

IN	Input
INJ	Injection
IO	Entrada e Saída (Input/Output)
kB	Quilobytes
Kd	Constante D do PID
kgfm	Quilograma-força x Metro
Ki	Constante I do PID
km	Quilometros
Kp	Constante P do PID
kPa	Quilopascal(is)
L	Litro(s)
LLSW	Low Level Software
MAF	Manifold Air Flow
MAP	Manifold Air Pressure
MB	Megabytes
MC	Measurement and Calibration
MDA	Measure Data Analyzer
MHz	Mega Hertz
mm	Milimetro
MPFI	Multi Port Fuel Injection
ms	Milisegundos
OSEK	Offene SystemeunddderenSchnittstellenfür die Elektronik in Kraftfahrzeugen
OUT	Output
PFI	Port Fuel Injection
PID	Proporcional, Integral e Derivativo
PWM	Pulse Width Modulation
rpm	Rotação por minuto
RTOS	Real-Time Operating System
SE	Software Engineering
SW	Software
TDC	Top Dead Center
TV	Test and Validation
V	Volt(s)

VHT	Very High Torque
VVT	Variable Valve Timing
VW	Volkswagen
WOT	Wide Open Throttle

SUMÁRIO

1. Introdução

1.1. Identificação das Necessidades

A indústria automotiva no Brasil, da mesma maneira que muitas áreas da engenharia brasileira, basicamente reproduzem e fabricam projetos desenvolvidos no exterior com poucas adaptações para o mercado brasileiro. Em particular o desenvolvimento de centrais eletrônicas automotivas (i.e. ECU) não foge deste padrão, apesar de apresentar um vasto trabalho e esforço local no que diz respeito à calibração de injeções eletrônicas.

No entanto, o trabalho de calibração (3ª etapa) é apenas uma das três principais etapas de todo o processo de desenvolvimento na indústria, que requer um trabalho local devido tanto aos ajustes quanto ao ambiente, componentes e, principalmente, combustíveis. Quanto às outras duas etapas, 1ª etapa: desenvolvimento de software e 2ª etapa: teste e validação virtual se restringem quase que exclusivamente a desenvolvedores externos, e muito disto se deve à falta de mão de obra especializada no Brasil.

Para gerarmos um avanço significativo na agregação de valor aos insumos desta indústria, é necessária uma intensa e drástica mudança no processo de criação, pesquisa e desenvolvimento aos quais somos submetidos hoje em dia, incluindo, principalmente, instituições de ensino. Um esforço por parte das indústrias e universidades no âmbito de incentivar o desenvolvimento nacional com qualidade e complexidade suficiente para impactar o andamento dos ciclos de produção atuais é de extrema importância para mudar este paradigma.

Para isso, é necessário desenvolver softwares de ECU, da mesma maneira que a indústria o faz, pois, a metodologia utilizada atualmente (*Model-Based Software Engineering*), permite uma modularidade de funções, utilização de sistemas operacionais em tempo real e componentes de software de aplicação desenvolvidos independentemente do hardware/processador utilizado. Devido a esses recursos e metodologia, os sistemistas (i.e. Bosch, Magneti Marelli) conseguem desenvolver novas funções rapidamente, o que é extremamente vital para acompanhar a demanda dinâmica do mercado.

Desta maneira, para a academia conseguir desenvolver novas funções e componentes de software (SW) que sejam relevantes e inovadores para a área, é preciso que o ambiente educacional também desenvolva software com os mesmos recursos e metodologias. Apenas assim, será possível que a pesquisa e o desenvolvimento universitário brasileiro inovem e incentivem o crescimento dessa 1ª etapa, que hoje é mínimo.

Para mudarmos este cenário, devemos capacitar engenheiros de forma a conseguirem evoluir até chegarem próximos ao nível de desenvolvimento estrangeiro, para que possam ser considerados aptos para começar a desenvolver projetos mais significativos provando sua capacidade neste segmento.

Uma forma de buscar esta meta está na pesquisa e desenvolvimento acadêmico no país, para que o conhecimento seja desenvolvido e disseminado entre os interessados em buscar esta capacidade de criação e desenvolvimento. Após os primeiros resultados, podemos buscar incentivos para criação de estrutura que possibilite novos projetos e pesquisas e assim, num processo recorrente, conseguindo mais incentivo a cada passo dado, até chegar num ponto onde a indústria considere os detentores deste conhecimento e responsáveis pelos projetos como capacitados para estas atividades.

1.2. Declaração dos Objetivos do Projeto

O objetivo deste projeto consiste em desenvolver um software que gerencie um motor à combustão interna de ciclo Otto à gasolina, controlando injeção, ignição e corpo de borboleta eletrônico. No entanto, para suprir as necessidades definidas, este projeto desenvolve esse software utilizando-se das metodologias: *Model-Based Software Engineering* e *V-Model* automotivo, além de respeito às normas automotivas como ASAM MCD-2 MC que permitirá um trabalho de calibração profissional com todos os recursos utilizados na indústria.

Os objetivos preliminares do projeto consistem em:

- Conseguir dar a partida no motor Volkswagen (VW) 1.6l e estabilizar a marcha lenta;
- Desenvolver componentes de software e calibrá-los suficientemente para vários pontos de operação e transientes, preocupando-se com a dirigibilidade, desempenho e conforto, e.g., evitando trancos;
- Realizar um trabalho de calibração com intuito de melhorar potência e torque gerado.

Culminando, por fim, em uma ECU de gerenciamento de motor, desenvolvida seguindo o *V-Model* e com *software* modular (componentes), dentro da norma ASAM MCD-2 MC.

1.3. Histórico e Trabalhos Correlatos

Uma injeção eletrônica atual consiste basicamente no gerenciamento da injeção de combustivel, geração da centelha dentro da câmara de combustão e também controle do corpo de borboleta, tudo isso de maneira sincronizada com o ângulo do motor. Controla-se a válvula borboleta seguindo a demanda requerida no pedal de acelerador, o cálculo do tempo e ângulo de injeção é realizado considerando fatores estequiométricos e por fim, determina-se o ângulo de avanço de ignição com base em rotação do motor e pressão do ar de admissão.

Tomando-se como base projetos anteriores (como o Otto II, no qual foi desenvolvido, na EPUSP, um gerenciamento de um motor VW 2.0L) limita-se a quantidade de sensores e atuadores necessários para o funcionamento completo do motor. Desta maneira, os conceitos de controle dos atuadores (i.e., controle de torque, referência de rotação no pedal e etc.) são preservados neste projeto, com o intuito de somar ao desenvolvimento fatores como: metodologias de desenvolvimento de software automotivo (*Model-Based Development*), inserir e implementar uma arquitetura de SW segmentada e modularizada, *V-Model* automotivo e norma ASAM MCD-2 MC, além de possibilidade de um trabalho de calibração.

2. Referencial Teórico

2.1. OSEK

OSEK (*Offene Systeme und derem Schnittstellen für die Elektronik in Kraftfahrzeugen*; português: "Sistemas abertos e suas interfaces para eletrônica automotiva") é um padrão e norma que especifica sistemas operacionais de tempo real (*Real-Time Operating System* - RTOS) embarcados, principalmente aplicado à indústria automotiva. Em particular, o RTA-OSEK se trata de uma implementação comercial do OSEK atualmente utilizado em sistemas *single* core gerenciando as funções *multitasking* do *Software* de Aplicação (AppSW) e *Basic Software* (BSW).

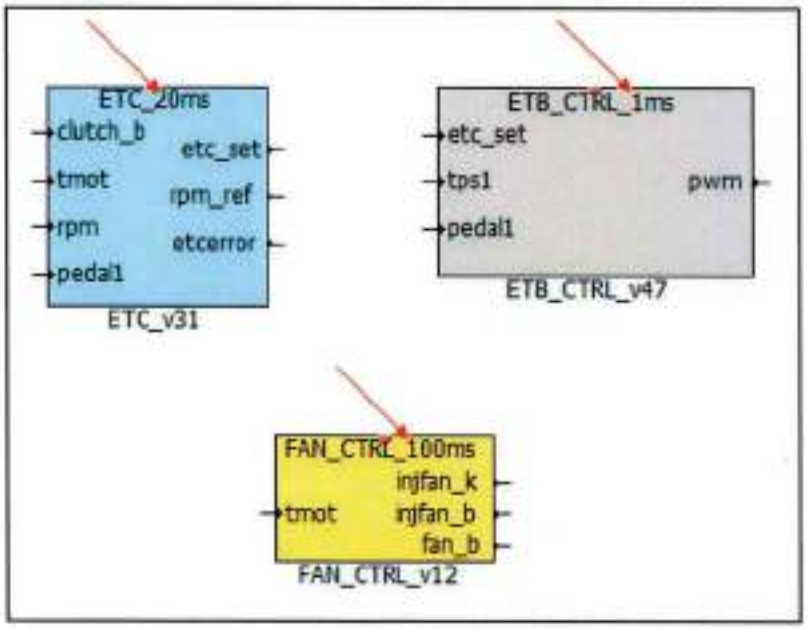
Por se tratar de um *single* core, este RTOS gerencia a execução dos processos internos do AppSW, resolvendo condições de disputa de processador de uma tal maneira que todos os processos rodam de forma quase paralela. Diversas ferramentas podem ser utilizadas para configuração, em alto nível, deste RTOS, de tal maneira, que a geração desta é feita de maneira automática para um *target* específico (microprocessador).

Por isso, é preciso associar no AppSW a execução de cada função à uma *task* do RTOS de tal maneira que esta será executada toda vez que o *trigger* da *task* for disparado. Entre os tipos de *tasks*, pode-se destacar *init* (primeira a rodar e apenas uma vez), *exit* (última a rodar e apenas uma vez) e *Alarm* (periódica e.g. 1ms, 10ms ou até mesmo síncrona – disparada periodicamente com a rotação do motor).

Assim, uma função associada à *task* de 10ms é disparada a cada 10ms, porém só é executada após a resolução das condições de disputa realizada pelo RTOS. Essas condições de disputa que se caracterizam por disparos simultâneos de duas *tasks* diferentes são resolvidos por prioridade e modo de *task* (preemptivo ou não-preemptivo).

Um exemplo pode ser observado na Figura 1 onde está destacada a periodicidade de execução de cada uma das funções mostradas. Observa-se que a função "ETC_v21" será disparada a cada 20ms, a "ETB_CTRL_v47" a cada 1ms e a "FAN_CTRL_v12" a cada 100ms, cabendo ao RTOS o disparo correto destas funções para que a CPU as execute (uma de cada vez).

Figura 1 – Exemplo de disparo associado à função

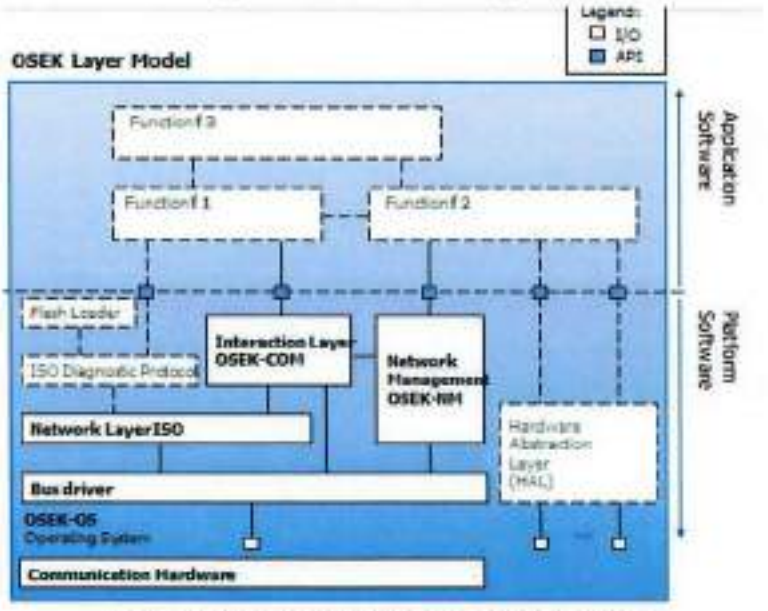


Fonte: o Autor

2.2. Código Objeto Autogerado

O desenvolvimento de SW automotivo baseia-se no conceito de *Model-Based* e sistema operacional real time conforme citado anteriormente e, para tal, é preciso uma plataforma de desenvolvimento que permita a geração de um código de maneira automática a partir de diagramas de blocos e configurações em alto nível do OS. Desta maneira, do ponto de vista de desenvolvimento, este foca-se no AppSW.

Figura 2 – Software Layout



Fonte: Automotive Software Engineering

2.3. Norma ASAM MCD-2 MC

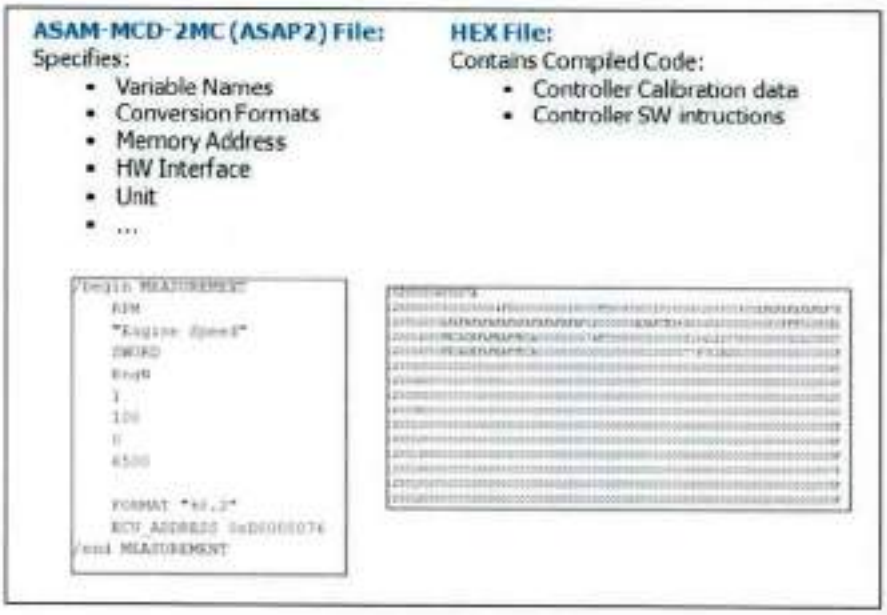
A norma ASAM MCD-2 MC consiste em um formato padronizado de medição e calibração automotiva criada pela ASAM (*Association for Standardizationof Automation and Measuring Systems*). A norma define a escrita de um arquivo de texto no formado *.a2l (A2L) no qual descreve a ECU e permite um trabalho de calibração utilizando ferramentas de calibração (MC *Tools*) padronizadas, de tal maneira que uma mesma MC *Tool* é capaz de acessar diversas ECUs, incluindo sistemistas diferentes, microcontroladores diferentes e etc.

O A2L é um arquivo que descreve o SW implementado na ECU, i.e., descreve variáveis de medição, parâmetros de calibração, protocolo de MC que a ECU suporta, segmentação de memória e outras informações. Por exemplo, no caso da variável RPM, o A2L contém: nome da variável (e.g. RPM), ponto da memória interna no qual encontra-se o valor dela, a unidade (e.g. 1/min), qual a fórmula de conversão de ponto fixo para unidades físicas (e.g. 2 bytes vão compreender de 0 a 8000 rpm) e se esta variável é um parâmetro de calibração ou medição (no caso do RPM seria medição, e no caso do fator proporcional do controlador PI, o P seria calibração).

ASAM MCD-2 MC determina a escrita deste arquivo A2L, porém, para um trabalho de calibração, é preciso também do código objeto a ser programado na ECU, i.e., é preciso também um arquivo *.hex (HEX - dependendo da tecnologia utilizada pode ter outros formatos, e.g. *.s19).

O HEX contém o código compilado em si (conjunto de instruções e estratégia compiladas do SW) e os valores dos parâmetros de calibração. Por exemplo, caso haja um mapa de avanço de ignição, no A2L estaria descrito um mapa de 6x7 sendo calibração, unidade em graus e formula de conversão, no entanto, os valores contidos neste mapa, estão no HEX.

Figura 3 – Exemplo de A2L e HEX



Fonte: o Autor

2.4. ECU *Bypass*

O processo de *bypass* nada mais é do que desviar a execução de uma rotina para outra sem a necessidade de editar o código fonte original e compilá-lo novamente. Este *bypass* pode ser realizado de diversas maneiras: interno (*On-Target*) ou externo; *bypass* de uma variável ou função inteira ou AppSW inteiro (*Fullpass*).

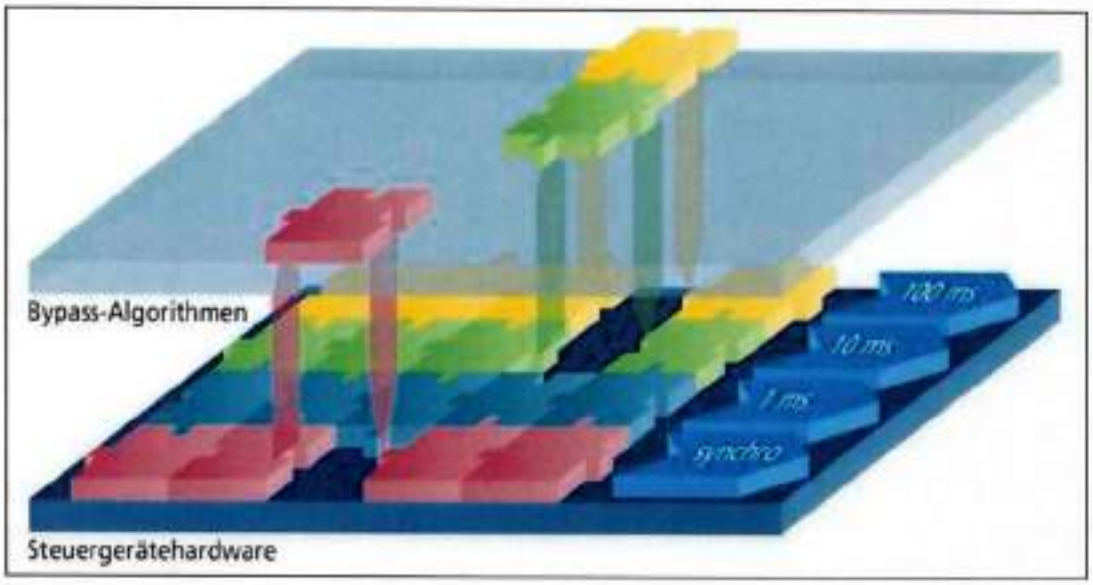
No caso de um *bypass* interno de uma função, pode-se chavear entre a função original e a de *bypass* através de um parâmetro de calibração. Este exemplo pode ser observado na Figura 4, onde *Function* (cinza) 1, 2 e 3 são as funções originais e a *Function* 1 (azul) é a de *bypass*.

Figura 4 – Exemplo de on-target bypass de uma função

Fonte: Bypass

De maneira geral, o AppSW roda sobre o RTOS (camada azul escura - Figura 4) e suas funções (representadas por peças de um quebra cabeça) podem ser desviadas e, ao invés de rodar a função original, executa-se as funções que estão na camada de *bypass* (cinza claro), e assim é possível, por exemplo, testar funções novas sem a necessidade de modificação do AppSW original.

Figura 5 – Execução de várias funções de bypass implementadas



Fonte: Bypass

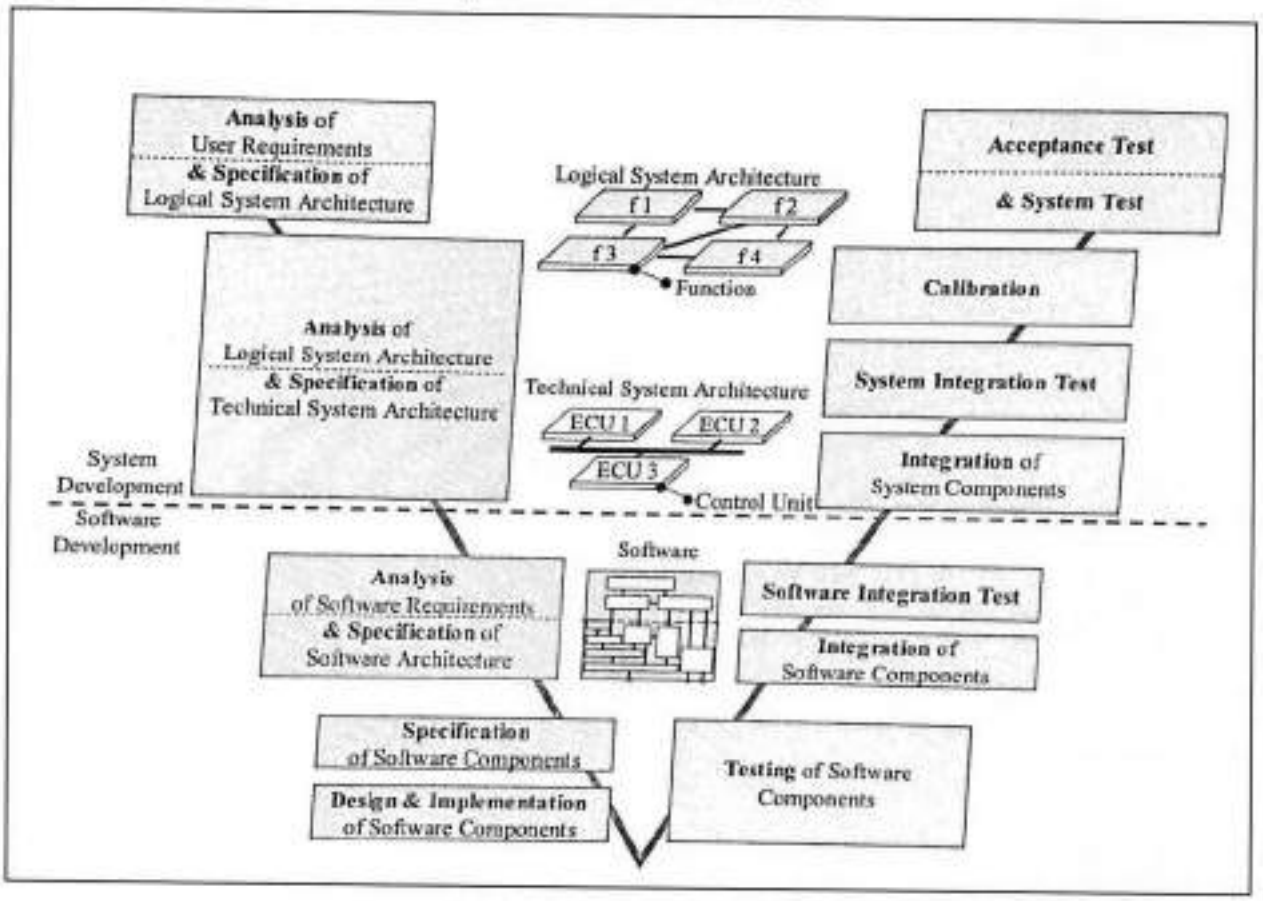
Esta camada de algoritmo de *bypass* pode estar presente dentro da ECU (chamado de *on-target*) ou pode ser executado por um HW externo dedicado chamado também de HW de prototipagem rápida. No caso deste projeto, foi utilizado *on-target bypass*.

3. Metodologia

A metodologia baseia-se no conceito *V-Model*, que consiste em etapas sequenciais, as quais guiam o desenvolvimento de um software automotivo e possibilitam uma validação robusta do AppSW.

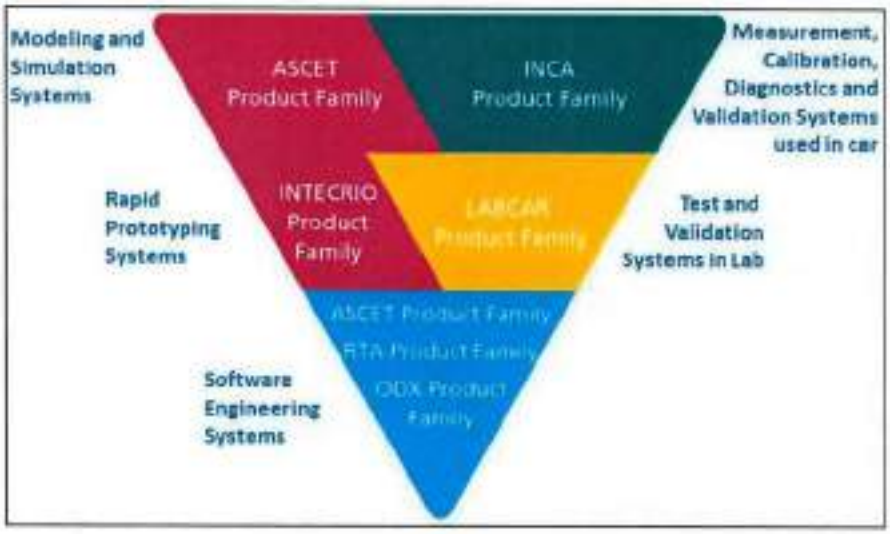
Abaixo na Figura 6 pode ser observado o V-model genérico de desenvolvimento de um sistema e, na Figura 7, encontra-se o V-model mais específico aplicado à este projeto com a inclusão das ferramentas utilizadas.

Figura 6 – V-Model Genérico



Fonte: *Automotive Software Engineering*

Figura 7 – V-Model aplicado ao projeto



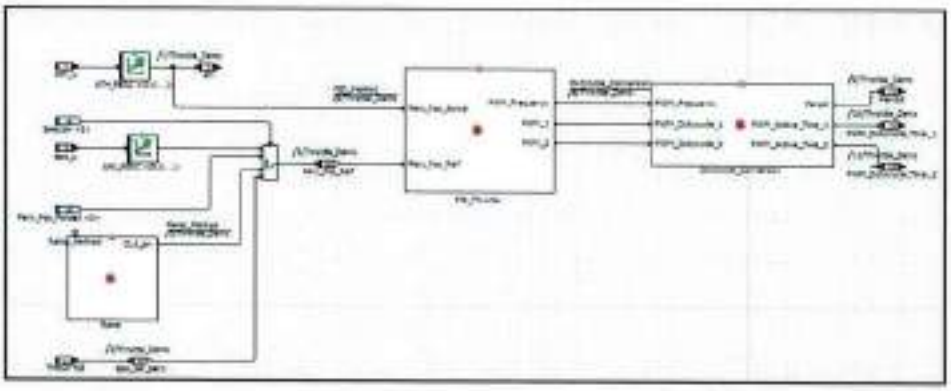
Fonte: Automotive Software Engineering

3.1. Control Design

Esta etapa é compreendida em vermelho na Figura 7 corresponde ao desenvolvimento do controle que será implementado na ECU, se importando inicialmente com o funcionamento lógico da estratégia. Desta maneira, o controle em si é desenvolvido em um gerador de funções que pode ser utilizado com as seguintes interfaces: diagrama de blocos, linguagem C, ESDL ou máquina de estados e, desta maneira, o código objeto será gerado automaticamente a partir destes diagramas.

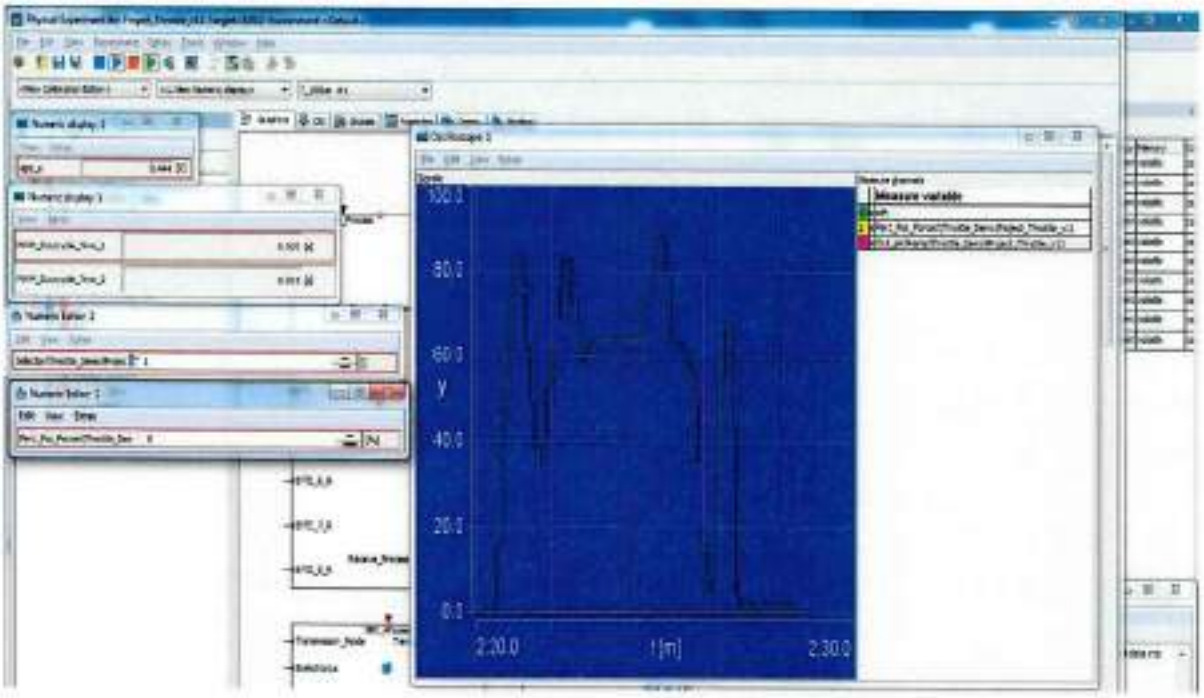
Os testes realizados nessa etapa são com entradas e saídas virtuais, podendo ser realizado em tempo real ou tempo virtual. No caso do primeiro (*real-time*), há a necessidade de um HW de prototipagem, que é um dispositivo dedicado para rodar o AppSW de controle e este, por ser dedicado, consegue rodar não só o AppSW como também o sistema operacional real time que será utilizado na ECU final, neste caso, um sistema operacional automotivo chamado RTA-OSEK por exemplo.

Figura 8 – Exemplo de controle desenvolvido no ASCET



Fonte: o Autor

Figura 9 – Exemplo de simulação com entradas e saídas virtuais no ASCET



Fonte: o autor

3.2. Software Engineering

Esta etapa se trata da geração do código objeto que será implementado na ECU, i.e., trata-se da compilação dos blocos criados na etapa anterior e a geração do código a ser programado. Nesta etapa, gera-se também dois arquivos muito importantes para a indústria automotiva o A2L e o HEX.

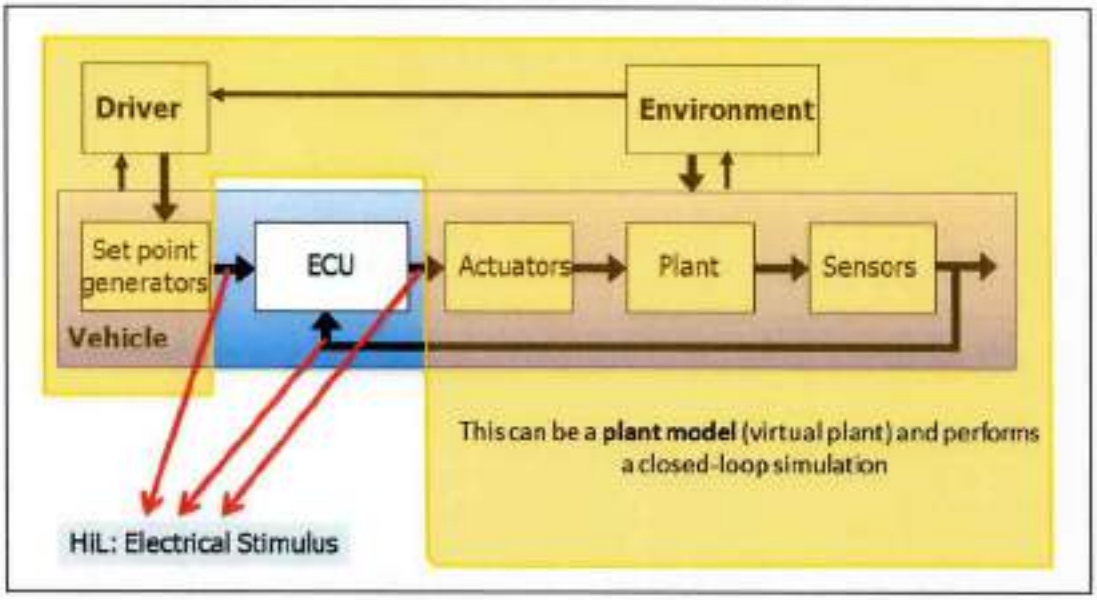
3.3. Virtual Test & Validation

Esta etapa consiste em gerar um modelo da planta e rodá-lo em um PC com sistema operacional real time conectado fisicamente à ECU, de tal maneira que a ECU rode em malha fechada com este modelo e permita um teste e validação completa do SW da ECU como se estivesse controlando a planta real.

No entanto, este *feedback* que a planta oferece para o controlador não se trata apenas de um valor ou uma variável, esta realimentação é feita de maneira elétrica, i.e., a ECU estimula o *Hardware-in-the-Loop* (HiL) com sinais elétricos reais (da mesma maneira que faria com a planta real) e o HiL (baseado em seu modelo) calcula quais seriam as saídas da planta considerando os estímulos da ECU e então gera a realimentação para a caixa de comando também de forma elétrica. E toda esta resposta é realizada em tempo real, de tal maneira que, para o referencial do controlador, é como se estivesse conectado ao sistema real.

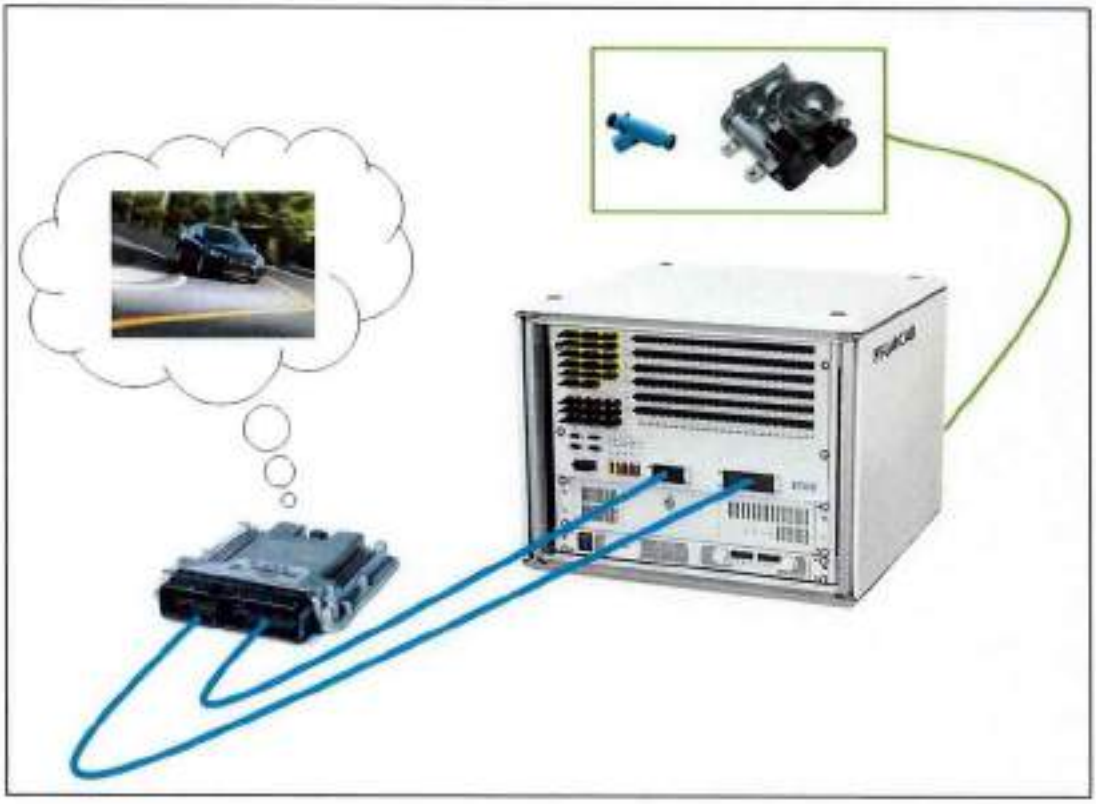
Para se obter um sistema completo, é possível até mesmo inserir no HiL modelos de plantas, atuadores, sensores, ambiente, e *setpoints* do motorista.

Figura 10 – Controle de Malha Fechada para HiL



Fonte: o Autor

Figura 11 – Sistema HiL



Fonte: o Autor

3.4. Measurement & Calibration

Última etapa do *V-Model* correspondente ao teste e validação na planta real, i.e., a ECU é conectada ao veículo real e o trabalho de calibração final dos parâmetros é realizado. A calibração é feita acessando a ECU através de protocolos automotivos de medição e calibração como, por exemplo, acesso via CAN CCP, ETK e etc.

Estes protocolos permitem um acesso à caixa de comando, recebendo os valores das variáveis de medição e também escrevendo valores de mapas e parâmetros de calibração online, i.e., alterando estes mapas enquanto a planta real (e.g. motor e transmissão) segue em funcionamento sem a necessidade de um novo flash da ECU. Nesta fase, ajustes relativos a combustíveis, sensores, dirigibilidade, economia de combustível, conforto e entre outros é realizada sem a necessidade de gerar e compilar um novo SW para isso.

3.5. Virtualização

Evidentemente que o trabalho de calibração não se resume apenas à quarta etapa citada. Este processo de ajuste dos parâmetros ideais para a ECU começa desde a primeira etapa, onde se desenha e define as funções que irão gerenciar o sistema. No entanto, quanto mais distante da planta real e de um teste em malha fechada, mais difícil fica definir os valores das variáveis de calibração e mais distantes estão estas dos valores que terão ao finalizar a etapa de MC. Assim, os problemas encontrados na última fase e que necessitem de uma solução de SW, precisam retornar para a etapa de *Control Design* e as mudanças passarem novamente pelo ciclo de desenvolvimento.

Portanto, o quanto antes o AppSW, o SW e a ECU forem testadas em *closed loop*, mais rápido se encontram e se resolvem os problemas. Além disso, é possível iniciar um trabalho de MC muito mais representativo desde a primeira etapa tornando o desenvolvimento de uma caixa de comando bem mais eficiente!

Considerando este conceito, o *V-Model* permite interações entre suas etapas através de diversos recursos como: *Software-in-the-Loop* (SiL), *Model-in-the-Loop* (MiL), Prototipagem Rápida, *External Bypass* e *On-Target Bypass*.

3.6. Model-Based Development

O *Model-Based Development* consiste em desenvolver um SW de maneira modular de tal forma que as funções e o AppSW de maneira geral se tornem independentes do HW. Desta maneira, o mesmo controle de corpo de borboleta pode ser integrado em outro AppSW de maneira imediata, sem necessidade de adaptação, desde que mantidas as mesmas entradas e saídas.

Todo esse desenvolvimento é possível graças ao SW automotivo ser baseado no RTOS automotivo (e.g. RTA-OSEK). Em níveis de abstração, o software só acessa o RTA-OSEK e o *Basic Software* (BSW) que contém os *drivers* (software, não hardware) de entrada e saída além de *drivers* complexos (e.g. injetor). Isso permite que o AppSW possa ser desenvolvido apenas enxergando o OS e os *drivers* e não o HW, tornando-o *target-independent* e modular.

4. Descrição do Projeto

4.1. Motor Volkswagen 1.6L MPFI

O motor a ser controlado pela ECU desenvolvida neste projeto se trata de um Volkswagen 1.6L TOTALFLEX 8V, montado em um Gol G5 2008. Apesar de o motor ser flex, o projeto foi desenvolvido em sua totalidade apenas com Gasolina. Além disso, o motor foi utilizado em sua configuração original, i.e. configuração geométrica original, sensores e atuadores originais, na Tabela 1 observam-se algumas características do motor:

Tabela 1 – Especificações do Motor

Motor	EA-111 VHT - álcool e/ou combustível
Cilindros	4 cilindros
Número de Válvulas	8v
Taxa de Compressão	12,1:1
Cilindrada	1598 cm^3
Potência Máxima	101cv @ 5250 rpm (G)
Torque Máximo	15,4 (kgfm) @ 2500 rpm (G)
Diâmetro x Curso	76,5 x 86,9 mm
ECU Original	Multiponto Bosch ME 7.5.30
Injeção	Multiponto Sequencial (MPFI) 1-3-4-2
Ignição	Acionamento banco a banco - 2 bobinas de ignição dupla acionando simultaneamente os cilindros 1-4 e 2-3 (centelha perdida)

Fonte: Ficha Técnica Volkswagen Gol G5 1.6L TOTALFLEX

4.2. Hardware

4.2.1. FlexECU

Neste projeto, foi utilizada uma FlexECU Gasolina, que se trata de uma ECU aberta, permitindo um desenvolvimento rápido e eficiente de novas estratégias de controle como foco principal no setor automotivo, uma vez que possui *drivers* específicos para sensores e atuadores automotivos. As especificações de HW da versão Gasolina podem ser observadas na Tabela 2.

Figura 12 – FlexECU

Fonte: Flyer FlexECU

Tabela 2 – Especificação da ECU utilizada no Otto III

FlexECU Gasoline– CCP		
ECU Core	*Microcontroller*	*Infineon Tri-Core TC1797 - 32bit*
Memory	*Clock Frequency*	*180MHz*
	Flash	*1.8 MB Code; 215kB Data*
	RAM	*43kB*
	EEPROM	*2kB (emulation)*
Inputs	*Supply Voltage*	*8 - 16 V*
	Digital Inputs	*8 (+2 dedicated to IGN and Flash)*
	Frequency Inputs	*4*
	Cam/Crankshaft Inputs	*Hall*
	Analog Inputs	*30 (11 passive and 19 active)*
	Lambda	*LSU and/or LSF*
	Knock	*Upto 4*
Outputs	*PWM Outputs*	*20 Low-Side; 1 High-Side*
	Digital Outputs	*11 Low-Side*
	H-Bridge Drivers	*3*
	Mass-Flow Valve Drivers	*2*
	Main Power Relays	*External*
	Injector Drivers	*Up to 8 (GDI or PFI)*
	Ignition Drivers	*Up to 8 (exteralcoil)*
Communication	*CAN*	*3 High Speed CAN*
MC Protocol	*Protocol*	*CCP - CAN Calibration Protocol*

Fonte: Flyer FlexECU

Todas as especificações da Tabela correspondem à componentes físicos de HW e não de controle. Ou seja, onde se lê GDI ou PFI, não quer dizer que a ECU tenha um controle GDI ou PFI e sim que há no HW os *drivers* necessários para comandar injetores para injeção direta e indireta.

A FlexECU não possui nenhum AppSW de gerenciamento de motor, possuindo apenas um *Low Level Software* (LLSW) que consiste em um RTOS + BSW que inclui funções para acionamento de alguns *drivers* e sincronismo. Por exemplo, o *Engine Position Management* (EPM) que consiste no sincronismo do sistema, i.e., recebe o sinal de *crankshafte camshaft* e com base na calibração realizada (ver detalhes em 5.3.1) consegue sincronizar todo o SW e acionar injetores e bobinas nos ângulos corretos. Um resumo dos *drivers* contidos no LLSW está disposto na Tabela 3.

Tabela 3 – Resumo de funções do LLSW

Nome	Categoria	Funcionalidade
Analog Input	Basic IOs	Leitura de sinais analógicos
Digital Input	Basic IOs	Leitura de sinais digitais
PWM Input	Basic IOs	Leitura de dutycycle e frequência de sinais
Digital Output	Basic IOs	Saída de sinais digitais
H-Bridge Output	Basic IOs	Saída para pontes H
PWM Output	Basic IOs	Saída de sinais de PWM (baixa potência)
Ignition Driver Output	*Drivers* Complexos	Gera o sinal sincronizado para ignição com base em argumentos inseridos
PFI Injector Driver	*Drivers* Complexos	Gera o sinal sincronizado para injeção PFI com base em argumentos inseridos
GDI Injector Driver	*Drivers* Complexos	Gera o sinal sincronizado para injeção GDI com base em argumentos inseridos
Lambda	*Drivers* Complexos	Controla o *heater* e opera uma sonda lambda narrow/wideband e lê o valor de lambda
Knock Detection Driver	*Drivers* Complexos	Detecta o *knock* com base em um circuito impresso com amplificador e filtros
Mass Flow Valve (MFV)	*Drivers* Complexos	*Driver* para controle da válvula de bombas de alta pressão de combustível
EPM	Sincronismo	Com base na calibração do *driver*, recebe sinais do *cam/crankshaft* e sincroniza o SW
VVT	Sincronismo	Correção no sincronismo caso trabalhe com VVT
CAN Protocol	Comunicação	Envio e recebimento de mensagens CAN (não envolve a comunicação CCP)
EEPROM	-	Escrita de valores relevantes na EEPROM durante o *running* da ECU

Fonte: Flyer FlexECU

Desta maneira, têm-se as seguintes interfaces:

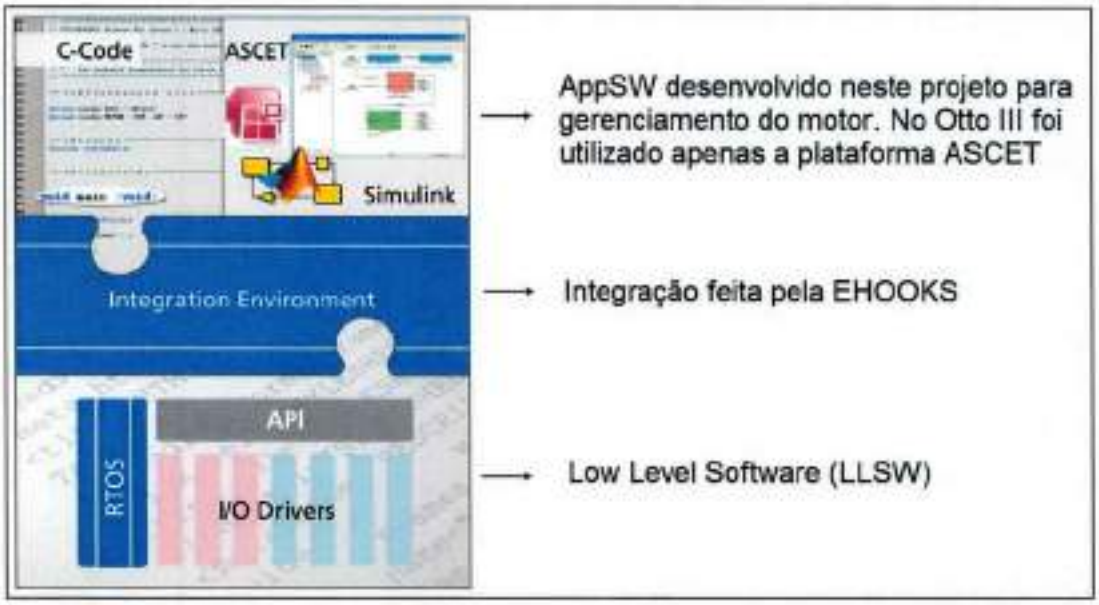
a. RTOS no qual o AppSW desenvolvido tem seus processos associados às *tasks* do sistema operacional;

b. Chamada de funções do LLSW via API dos *drivers*, *call* realizado pelo AppSW

c. Escrita e leitura valores do LLSW pelo AppSW como, por exemplo, a rotação atual calculada pelo EPM.

Para a integração das interfaces mostradas, é utilizado o software chamado EHOOKS (ver mais detalhes em 4.3.3). Desta maneira, o LLSW que totaliza BSW + RTOS é representado graficamente na Figura 13:

Figura 13 – Arquitetura de SW da FlexECU



Fonte: Flyer FlexECU

4.2.2. ES581.4

A FlexECU utilizada possui um protocolo de comunicação para MC chamado CCP (*CAN Calibration Protocol*), i.e., através da CAN 1 da ECU é possível realizar um acesso às variáveis de medição e parâmetros de calibração e alterá-los online. Para isso, são necessários 4 principais fatores (para qualquer acesso via CCP):

a. BSW da ECU com implementação do protocolo CCP;

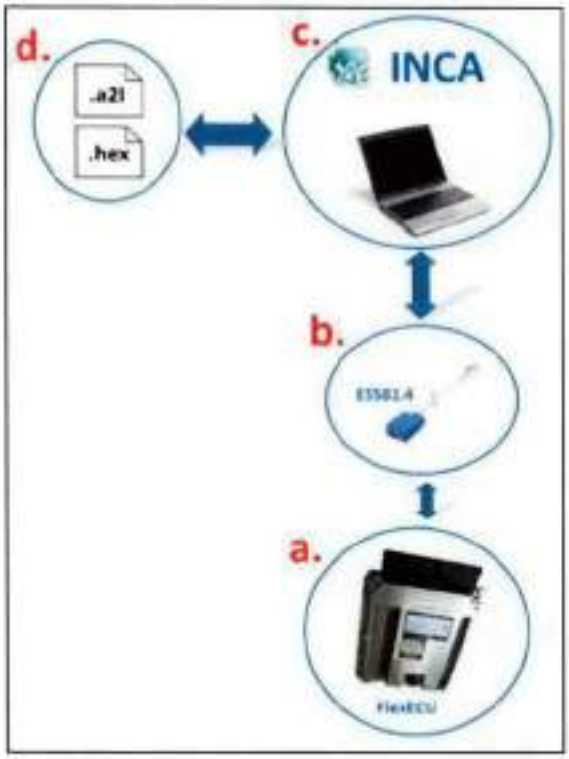
b. Um HW de acesso à ECU;

c. MC *Tool* – Aplicativo no PC que possibilite o calibrador realizar a tarefa de calibração com uma interface de usuário que permita um acesso em alto nível (e.g. não sendo necessário enviar manualmente todas as mensagens CAN em hexadecimal);

d. Arquivos da norma ASAM 2MC (A2L e HEX) que permitem o MC *Tool* acessar a ECU.

No caso do HW de acesso à ECU, como se trata de uma comunicação CCP baseada no protocolo CAN, foi utilizado o ES581.4 que é uma interface CAN-USB compatível com a MC *Tool* utilizada (INCA). Desta maneira, para atender os 4 requisitos, o *setup* desenvolvido para realização da etapa de MC foi o mostrado na Figura 14:

Figura 14 – Setup para MC



Fonte: o Autor

4.2.3. Medição de Lambda

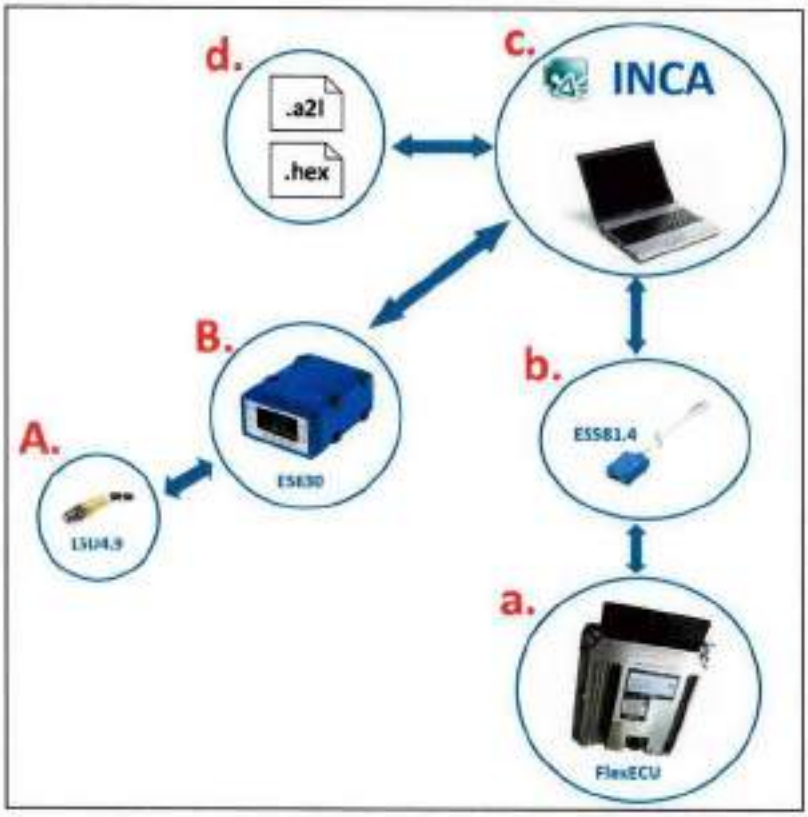
Para a etapa de medição e calibração e, principalmente para o levantamento dos mapas básicos de injeção e ignição, foi necessária a medição do valor de lambda de maneira precisa. A ECU original utiliza uma sonda lambda conhecida como *narrowband*, i.e., devido sua curva de sensor ser acentuada na região de lambda igual a 1, o valor medido é apenas um indicativo de mistura (rica ou pobre) não podendo, portanto, ser utilizado na etapa de desenvolvimento do novo controlador.

Para realizar a medição linear de lambda, foi necessário utilizar uma sonda lambda *wideband* e um lambda meter. Neste caso, foram utilizados:

- Sonda Lambda: Bosch – LSU4.9
- Lambda Meter: ETAS - ES630

Como neste projeto o controle de malha fechada de lambda não será desenvolvido, a LSU foi conectada diretamente no ES630 e este valor transmitido para o MC *Tool*, com intuito de visualizar e gravar o valor apenas para referência. Assim, o setup final de HW para o desenvolvimento do projeto está representado na Figura 15:

Figura 15 – Setup final para a etapa de MC



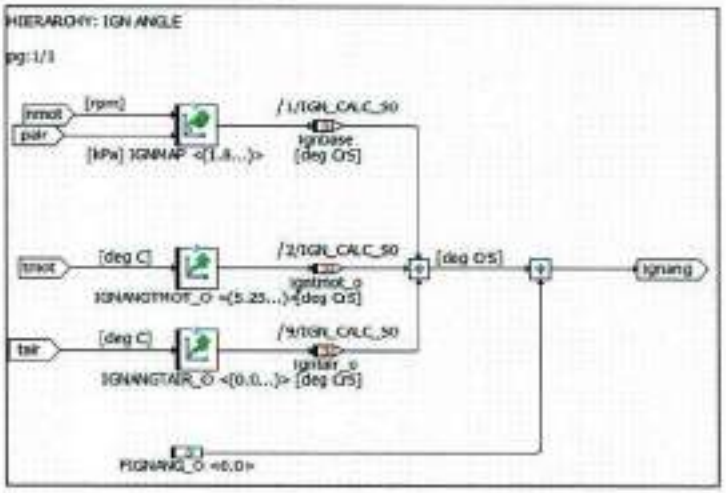
Fonte: o Autor

4.3. Software

4.3.1. ASCET

Utilizado desde 1997, e tendo atualmente mais de 68 milhões de ECUs com SW embarcado desenvolvido com esta ferramenta, o ASCET se trata de um gerador de funções com foco em geração de SW embarcado que permite o desenvolvimento em *Model-Based* do AppSW e geração automática de código objeto. O ASCET foi desenvolvido especialmente para a indústria automotiva, i.e., atende requerimentos do desenvolvimento de SW embarcado automotivo, como por exemplo, RTA-OSEK, geração de A2L e etc.

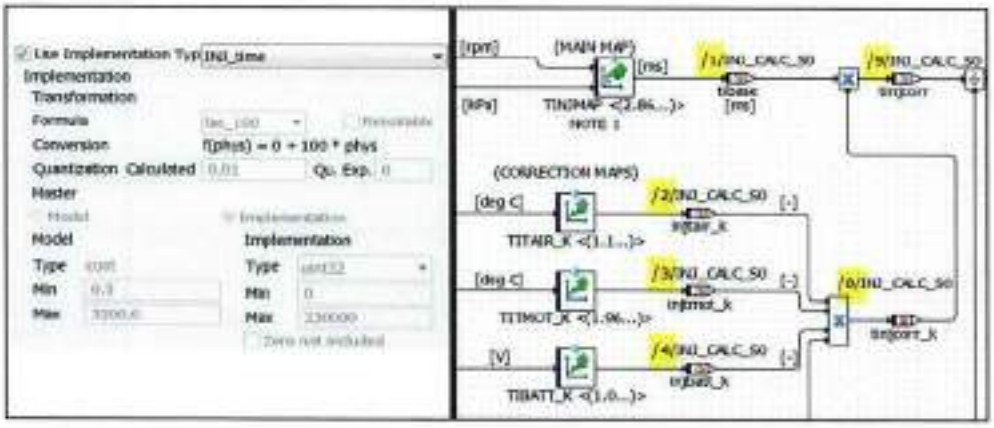
Figura 16 – Exemplo de diagrama de blocos desenvolvido no ASCET



Fonte: o Autor

Por se tratar de um gerador de funções com intuito principal de geração de código objeto para sistemas embarcados, o ASCET possui vários recursos importantíssimos para a geração automática do código. Entre estes recursos, podemos destacar a implementação de variáveis com ponto fixo e sequência de execução da função (como um código C será gerado a partir do diagrama de blocos, a sequência de execução é necessária para ordenar as linhas do código).

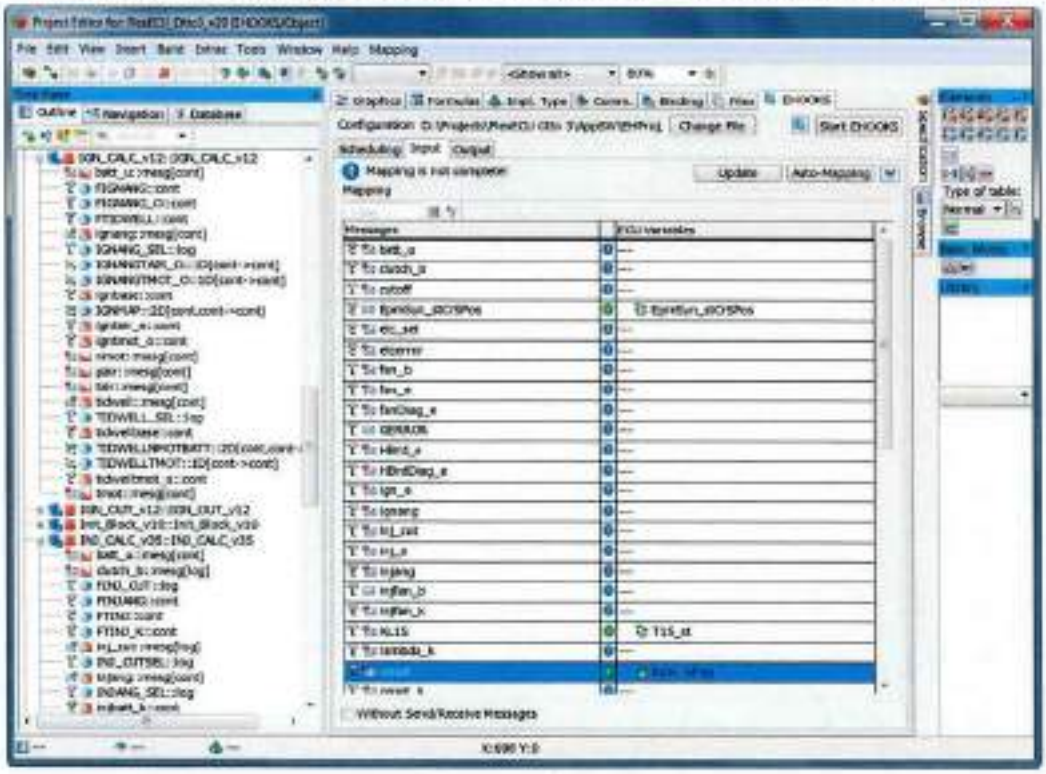
Figura 17 – Configuração do ponto fixo e sequência de execução



Fonte: o Autor

Neste projeto, o ASCET foi utilizado para desenvolvimento do AppSW, no entanto, por se tratar da FlexECU e um *on-target bypass*, a conexão entre LLSW e AppSW desenvolvido é feita pelo EHOOKS (ver detalhes em 4.3.3). A interação entre ASCET e EHOOKS é feita dentro do próprio ASCET, onde se determina o EHOOKS como *target* do sistema. Todas as associações são feitas no próprio ASCET que chama o EHOOKS em *background* quando o *build* do SW é realizado.

Figura 18 – Configuração e conexões do EHOOKS dentro do ASCET



Fonte: o Autor

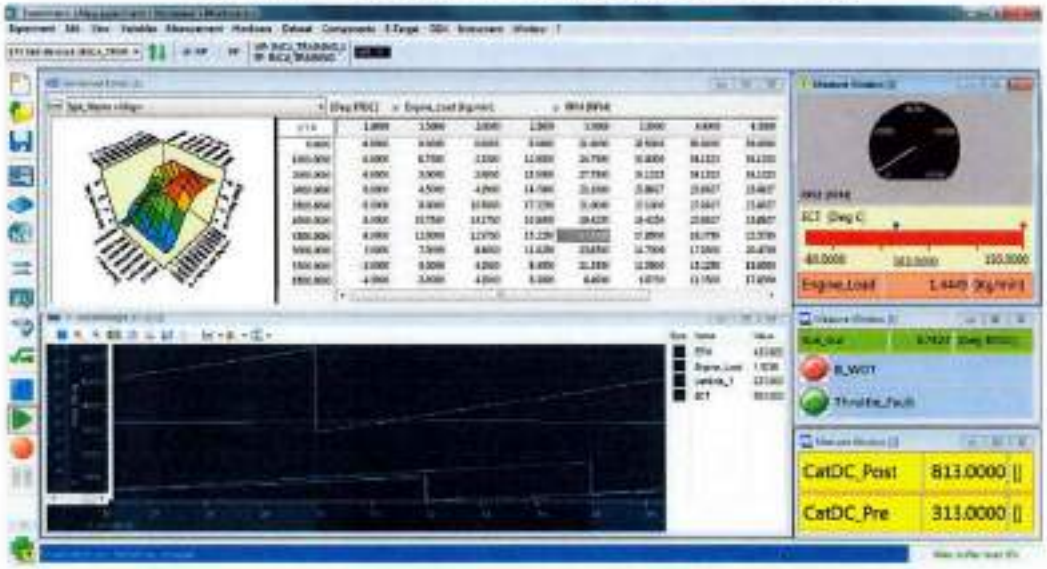
4.3.2. INCA

INCA é uma MC *Tool* amplamente utilizada na indústria automotiva para medição e calibração em banco de motores, dinamômetros de rolo, HiL, veículos, etc. Esta ferramenta oferece diversos recursos necessários durante um desenvolvimento de ECU, entre eles:

- Flash da ECU;
- Configuração de HW (e.g. medição de lambda na mesma base de tempo das variáveis internas da ECU);
- Medição e Calibração da ECU;
- Armazenamento de dados das variáveis da ECU e HW externo;
- Pós análise de dados gravados (*Measure Data Analyser* – MDA);
- Gerenciamento de calibrações (comparação entre *datasets* de calibração).

Para acessar a ECU, (conforme comentado na em 2.3) é preciso que o SW seja desenvolvido de acordo com o padrão ASAM MCD-2 MC, i.e., gerar os arquivos A2L e HEX e inseri-los no INCA. Feito isso, com o HW de acesso à ECU correto (no caso do projeto o ES581.4) é possível realizar um desenvolvimento de MC.

Figura 19 – Printscreen do Experimento do INCA



Fonte: o Autor

4.3.3. EHOOKS

EHOOKS é uma ferramenta de software que facilita a utilização e inserção de *bypass* na da ECU. Incluindo apenas o A2L e HEX é possível inserir os *hooks* e as funções de *bypass* mesmo sem ter o código fonte original ou qualquer detalhe do SW que originou o HEX inserido.

No caso deste projeto, a FlexECU contém o LLSW que é fornecido através dos arquivos A2L e HEX (i.e. sem acesso ao código fonte do LLSW). Por isso, para inserir um novo AppSW que irá gerenciar o motor, é preciso inseri-lo como funções de *on-target bypass* associadas às *tasks* do RTOS que já estão implementado no HEX original da FlexECU.

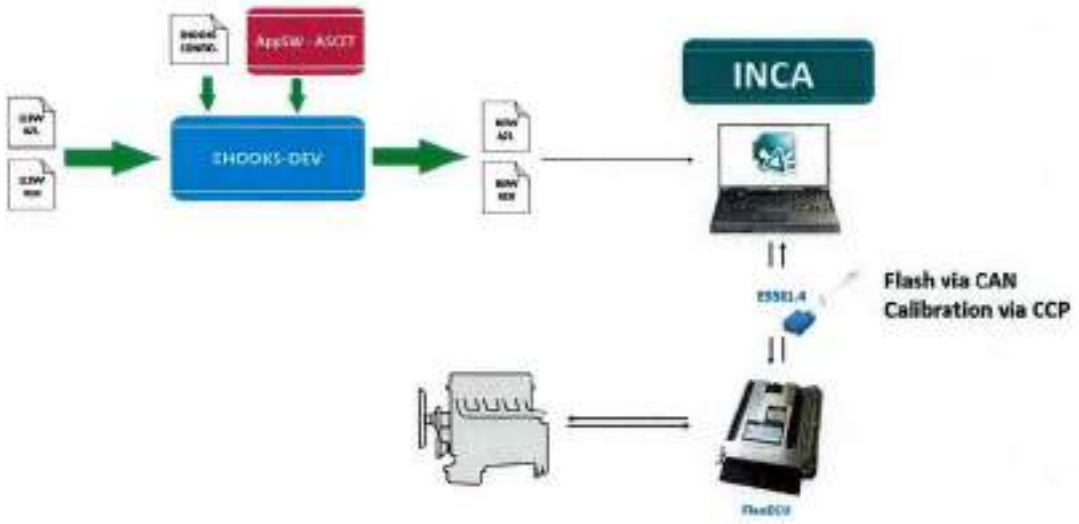
Desta maneira, o workflow da inserção deste AppSW inclui:

- Configuração de EHOOKS
- A2L e HEX originais do LLSW
- AppSW do ASCET

Assim, o build do EHOOKS resulta em um novo A2L e HEX com base nas associações entre processos do AppSW e *tasks* do RTOS, como pode ser observado na Figura 20:

4.4. Workflow completo do desenvolvimento

Figura 20 – Workflow do desenvolvimento da ECU



Fonte: o Autor

Portanto, o AppSW é desenvolvido no ASCET, onde associa os processos às *tasks* do LLSW. Com o A2L e HEX originais do LLSW e o AppSW o EHOOKS gera um novo A2L e HEX. Estes novos arquivos são inseridos no INCA, onde é realizado o flash da ECU com esse novo HEX. Após o Flash, é possível realizar os testes das novas funções na planta real e realizar a calibração através do INCA e do HW de interface, no caso o ES581.4.

5. Desenvolvimento do gerenciamento do motor

5.1. Aplicação da Metodologia

O principal diferencial deste projeto em relação aos anteriores jaz no uso da metodologia automotiva *V-Model*, cobrindo todas as etapas possíveis dada a estrutura disponível, o que restringiu, apenas, a execução do *Virtual Test &Validation*.

Iniciando pela etapa mais essencial do desenvolvimento do controle, as funções foram projetadas de uma forma abstrata e lógica para que fosse possível pensar nos detalhes do funcionamento e posteriormente implementadas utilizando a plataforma ASCET.

Esta implementação compreende o desenho dos blocos e interligações, atribuições computacionais, i.e., diferentes tipos de variáveis, frequência de execução, etc., testes utilizando simulação virtual e HW de prototipagem rápida. A simulação virtual visa validar o funcionamento matemático do bloco, além de proporcionar uma visão da resposta do controle, permitindo uma avaliação em relação à pretendida no desenvolvimento. Após esta fase, prosseguimos para o teste utilizando os HW de prototipagem rápida ES910 e ES930. Este HW permite que, após o seccionamento dos sinais entre a ECU original e o carro, sejam testadas individualmente cada função com sinais reais, uma vez que este HW consegue rodar apenas esta função sem depender do conjunto todo, o que facilita a identificação precoce de erros no controle bem como a visão de melhorias.

A segunda etapa de *Software Engineering*, que se trata da geração do A2L e HEX, foi realizada de maneira automática pelo *auto-code generator* do ASCET que ainda chama o EHOOKS em background para finalizar a compilação do código realizando a interface entre LLSW e AppSW desenvolvido no ASCET.

Para a última etapa, a de medição e calibração, foi feito o uso de um dinamômetro de chassis para a parte inicial deste processo. Melhor detalhada posteriormente, esta fase consistiu em cálculos básicos para os parâmetros de calibração, que permitiu partir o motor e manter marcha lenta. Quase 100 horas de testes no dinamômetro de chassis foram necessários para:

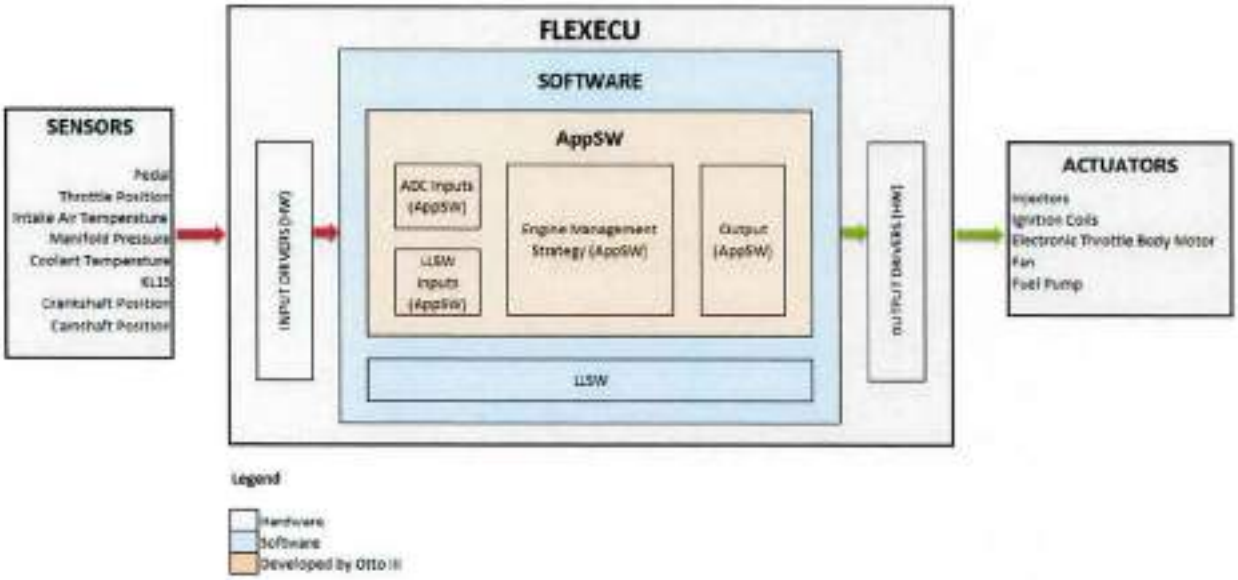
- Levantar os mapas básicos de injeção e ignição ajustados ponto a ponto de operação;
- Calibração do controle de rotação;
- Testes dinâmicos inserindo carga ao dinamômetro e simulando situações reais de rua;
- Simulação de possíveis falhas e erros para conferencia do funcionamento esperado das estratégias de *safety*.

Assim que os resultados se mostraram consistentes, com um funcionamento robusto e adequada calibração relativa ao controle de rotação foi iniciado a etapa de *road test* para calibração relativa à dirigibilidade, conforto e refino dos mapas, totalizando em mais de 500 km de testes.

5.2. Overview do Sistema

O gerenciamento do motor consiste em basicamente ler os valores dos sensores e através de estratégias de gerenciamento controla-se os atuadores do sistema. Para isso são precisos componentes de HW e SW mostrados em uma visão holística do sistema na Figura 21:

Figura 21 – Overview do Sistema

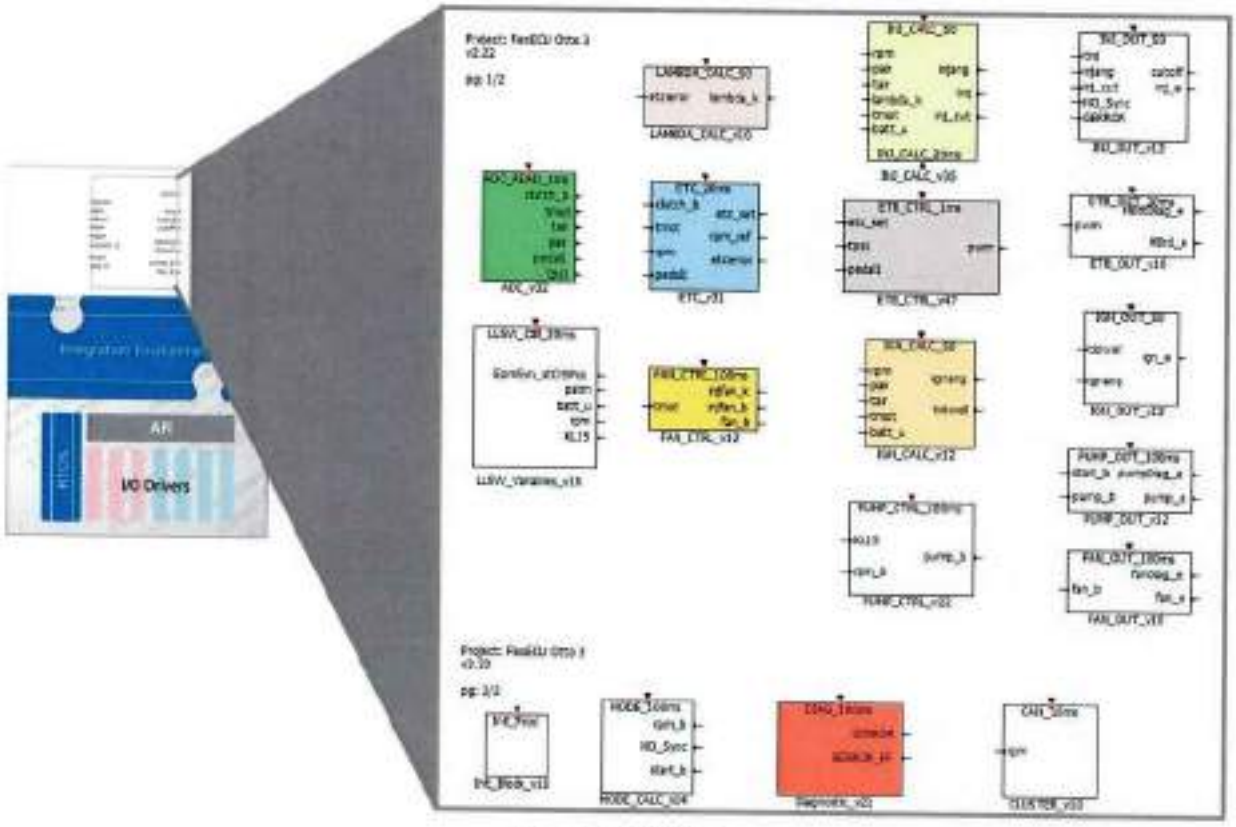


Fonte: o Autor

Neste projeto foram considerados apenas os sensores e atuadores mostrados na Figura 21. Além disso, a interface entre os sensores, atuadores e Software (AppSW + LLSW) é feito pelos *drivers* (HW) da FlexECU.

Com relação ao SW, ele é mostrado em mais detalhes na Figura 22, no qual é possivel observar o nível 0 do AppSW. Para o projeto Otto III foi preciso calibrar o LLSW para os sinais dos sensores e atuadores do VW 1.6, depois desenvolver todo o AppSW para gerenciamento do motor e por fim calibrá-lo.

Figura 22 – Detalhamento do SW



Fonte: o Autor

5.3. Low Level Software

Como explicado em 2.1 a FlexECU contém um LLSW que compreende o RTA-OSEK e o BSW que contém funções simples como o sincronismo. Dentre todos estes recursos oferecidos pelo LLSW, neste projeto foram utilizados apenas alguns deles, destacados na Tabela 4:

O maior detalhamento de cada um dos componentes do LLSW se encontra no manual da FlexECU, no entanto, o PFI e o EPM possuem uma configuração um pouco mais complexa que é extremamente relevante para o projeto e por isso serão abordadas a seguir.

Tabela 4 – Componentes do LLSW utilizados no Otto III

Nome	Função API	Entrada	Saída
Analog Input	DevLib_Adc_Get	Pino	Tensão no Pino Diagnostico
Digital Output	DevLib_Dio_Set	Pino Valor	Diagnostico
H-Bridge Output	DevLib_HBridge_Out	Pino Dutycycle Periodo Enable ...	Diagnostico
Ignition Driver	IgnCl_PfiSetParam	Ângulo de Ignição Dwell Time Enable ...	Diagnostico
PFI Injector Driver	InjVlvPfi_SetParam	Ângulo de injeção Tempo de injeção Cutoff ...	Diagnostico
CAN Protocol	Documentação da FlexECU		
EPM	Detalhes a seguir		

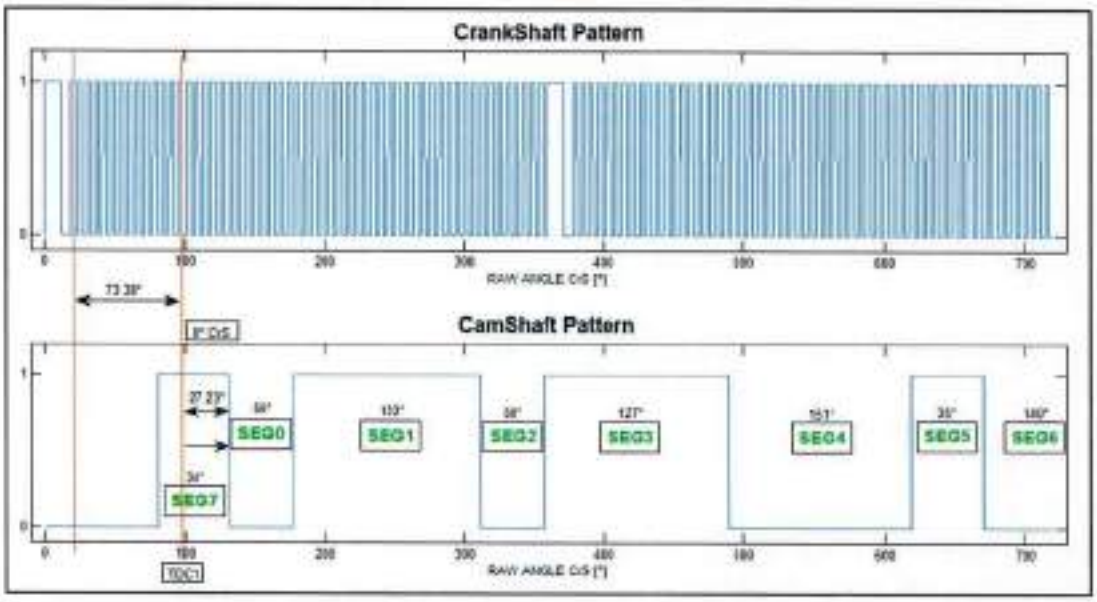
Fonte: o Autor

5.3.1. *Engine Position Management* (EPM)

O EPM é um componente responsável pelo sincronismo da ECU, depois de configurado, ao receber os sinais do sensor de *camshaft* e *crankshaft* (ambos sensores de efeito Hall) é possível saber o ângulo em que o motor está e assim acionar os injetores e as bobinas no momento correto e sincronizado. Além disso, é possível obter deste módulo do LLSW variáveis importantes como rotação e ângulo do motor.

A configuração do EPM é a caracterização, através de parâmetros de calibração, do padrão de sinal entre o *camshaft* e o *crankshaft* (Figura 23), i.e., é preciso informar um conjunto de 12 variáveis de calibração (algumas são vetores) que permitem o LLSW entender este padrão e assim sincronizar com os sinais de entrada.

Figura 23 – Padrão de sinais de roda fônica e sensor de fase



Fonte: o Autor

O intuito então é caracterizar estes dois sinais dentro da FlexECU através de alguns parâmetros de calibração, o detalhamento deste processo se encontra no manual da FlexECU. Na Tabela 5, foram destacados os valores utilizados no projeto Otto III, que corresponde à caracterização do padrão da Figura 24.

Considerando estes parâmetros, ressalta-se a função de alguns deles como:

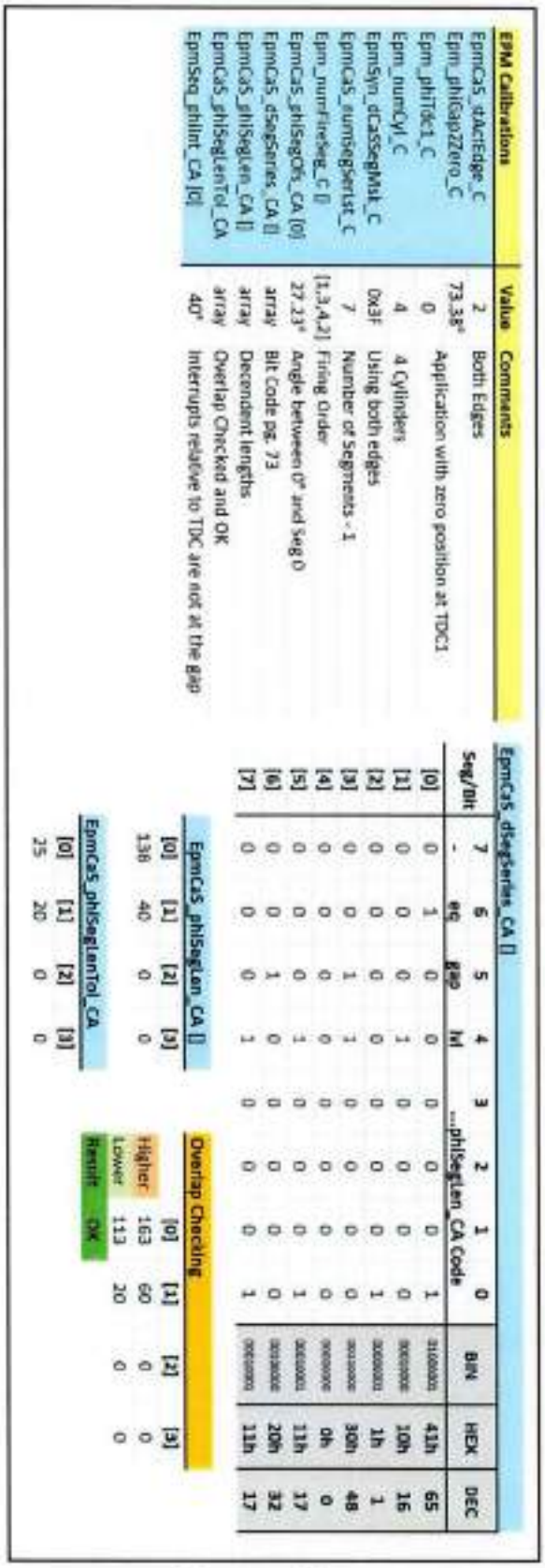
Tabela 5 – Calibrações do EPM relevantes

Epm_phiGap2Zero_C	73.38°	Trata-se do angulo entre a falha e a referência 0° - no caso TDC1
Epm_phiTdc1_C	0	Esta variável com valor zero define o TDC1 como referência de 0°
Epm_numCyl_C	4	Configura-se que são 4 cilindros
Epm_numFireSeg_C []	[1,3,4,2]	Ordem de ignição
EpmSeq_phiInt_CA [0]	40°	Angulo no qual a interrupção S0 ocorrerá, no caso 40° antes do TDC1

Fonte: o Autor

O S0 é uma interrupção síncrona, ou seja, os processos associados a esta *task* são disparados toda vez que o motor passar por ângulos específicos. Desta maneira, esta *task* apesar de periódica não possui um período constante, pois o seu *trigger* está associado à rotação do motor. Esta *task* síncrona é extremamente importante para variáveis relevantes à injeção e ignição (neste projeto Otto III injeção e ignição são calculadas nesta *task* S0).

Figura 24 – Parâmetros de Calibração do EPM para sinais do Gol



EPM Calibrations	Value	Comments
EpmCas_dActEdge_C	2	Both Edges
Epm_phiGap2Zero_C	73.38°	
Epm_phiTdc1_C	0	Application with zero position at TDC1
Epm_numCyl_C	4	4 Cylinders
EpmSyn_dCasSegMsk_C	0x3F	Using both edges
EpmCas_numSegSeries_C	7	Number of Segments - 1
Epm_numFireSeq_C []	[1,3,4,2]	Firing Order
EpmCas_phiSegOfs_CA [0]	27.23°	Angle between 0° and Seg 0
EpmCas_dSegSeries_CA []	array	Bit Code pg. 73
EpmCas_phiSegLen_CA []	array	Decendant lengths
EpmCas_phiSegLenTol_CA	array	Overlap Checked and OK
EpmSeq_phiInt_CA [c]	40°	Interrupts relative to TDC are not at the gap

EpmCas_dSegSeries_CA []											
Seg/Bit	7	6	5	4	3	2	1	0	BIN	HEX	DEC
	-	eq	gap	hl	...phiSegLen_CA Code						
[0]	0	1	0	0	0	0	0	1	01000001	41h	65
[1]	0	0	0	1	0	0	0	0	00010000	10h	16
[2]	0	0	0	0	0	0	0	1	00010001	1h	1
[3]	0	0	1	1	0	0	0	0	00110000	30h	48
[4]	0	0	0	0	0	0	0	0	00000000	0h	0
[5]	0	0	0	1	0	0	0	1	00010001	11h	17
[6]	0	0	1	0	0	0	0	0	00110000	20h	32
[7]	0	0	0	1	0	0	0	1	00010001	11h	17

EpmCas_phiSegLen_CA []			
[0]	[1]	[2]	[3]
136	40	0	0

Overlap Checking				
	[0]	[1]	[2]	[3]
Higher	163	60	0	0
Lower	113	20	0	0
Result	OK			

EpmCas_phiSegLenTol_CA			
[0]	[1]	[2]	[3]
25	20	0	0

Fonte: *o Autor*

5.3.2. *Driver* de injetor PFI

O *driver* de PFI, depois de configurado, consiste, neste projeto, em receber o tempo de injeção, ângulo de injeção e *cutoff* e acionar de maneira sincronizada os injetores com estes valores. No entanto, para restar apenas estes valores, outros parâmetros foram definidos previamente para configurar o *driver*. A Figura 25 facilita a visualização e compreensão do funcionamento do *driver*.

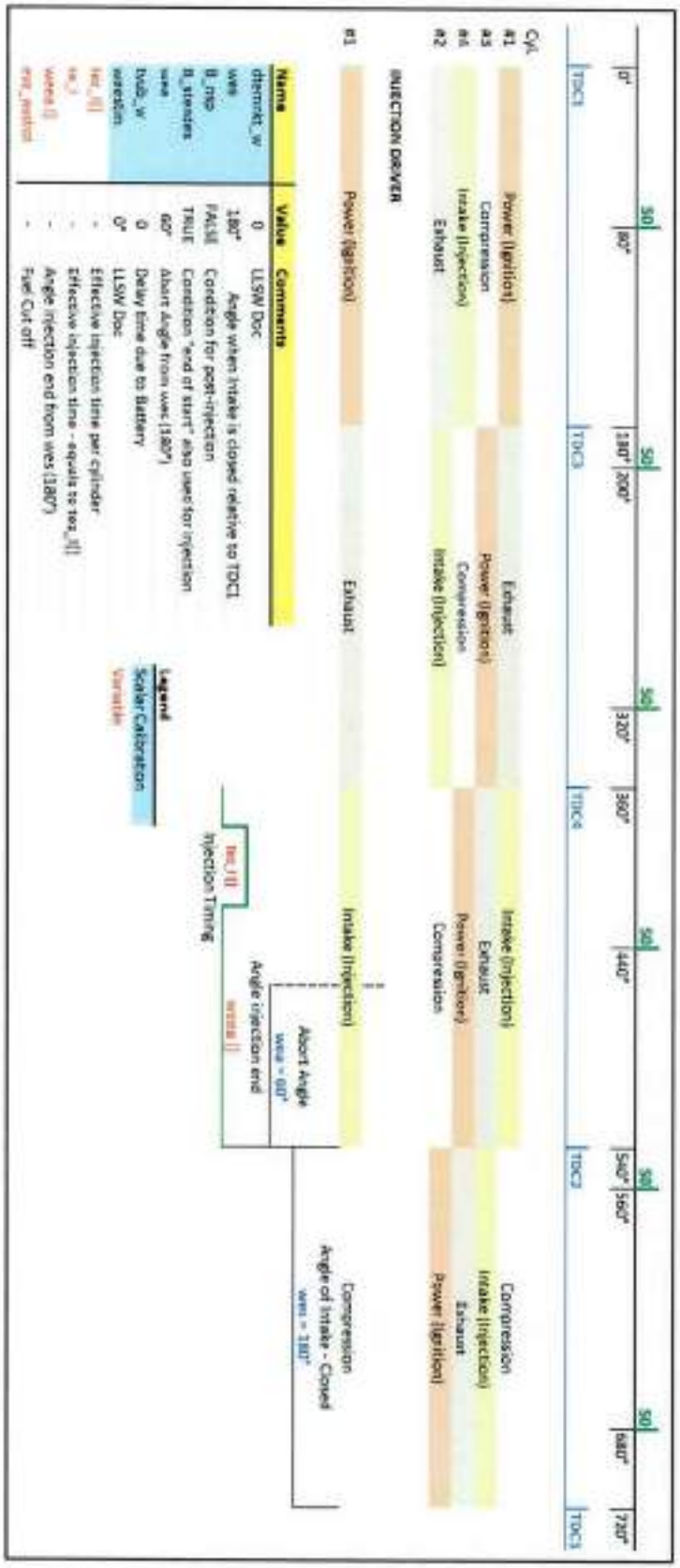
Na parte superior da Figura 25 tem-se o eixo preto de 0 a 720°CrS e como no EPM foi definido o TDC1 como 0°CrS o eixo em azul com os respectivos *Top Dead Center* (TCD – Ponto Morto Superior). Também foi definido no EPM que o S0 (*task* - interrupção - síncrona) está à 40° do TDC1 e repete-se o *trigger* a cada 120°. O S0 está representado em verde no eixo preto.

Em seguida, considerando a ordem de ignição (1-3-4-2) do motor VW utilizado, observa-se a etapa do ciclo de 4 tempos que cada um dos cilindros se encontra no decorrer do eixo preto. Em um ciclo de 4 tempos, os valores de tempo e ângulo de injeção, *cutoff*, *dwell* time e ângulo de ignição são calculados apenas na *task* S0, ou seja, apenas uma vez a cada ciclo.

No caso da injeção, a referência é do TDC para trás, ou seja, define-se o valor de "wes" que se trata de quantos graus antes do TDC a válvula de admissão está fechada (no caso 180°). Além disso, define-se um ângulo de aborto ("wea", no projeto 60° - com relação ao "wes"), ou seja, depois deste ângulo, nenhum combustível será injetado mesmo que ainda tenha tempo de injeção restante, pois injetar muito próximo do fechamento da válvula não surte efeito, dada a possibilidade de não dar tempo de o combustível ser admitido na câmara neste ciclo. Ambos os parâmetros podem ser observados em mais detalhes na Figura 26.

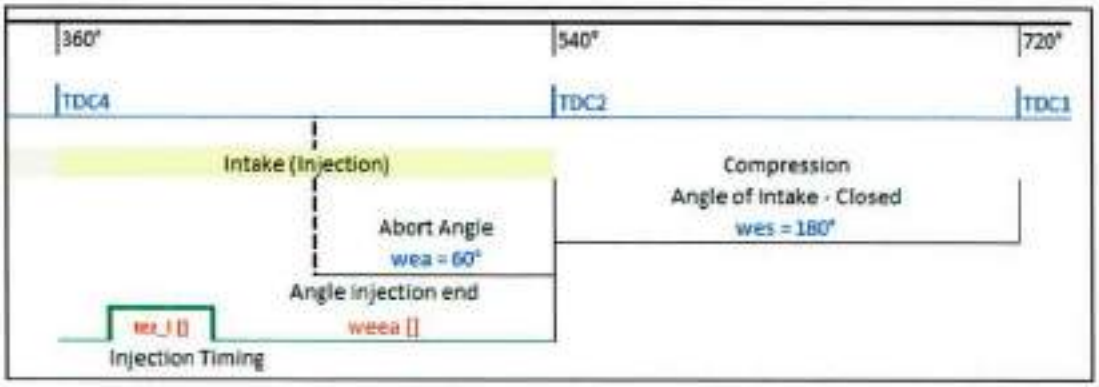
Em verde pode-se observar como seria o sinal de comando do injetor, considerando o "wes" e o "wea". O ângulo de injeção é caracterizado pela variável "weea" com referência ao "wes" e o tempo no qual o injetor fica acionado é definido pela variável "tez_I". No caso do AppSW, não foram utilizados estes nomes para as variáveis, para facilitar o entendimento tempo de injeção é "tinj" e ângulo de injeção "injang", e esses valores são passados para essas variáveis do LLSW em um último momento.

Figura 25 – Configuração e funcionamento do driver PFI



Fonte: *o Autor*

Figura 26 – Detalhamento do driver de PFI



Fonte: o Autor

5.4. AppSW - Gerenciamento

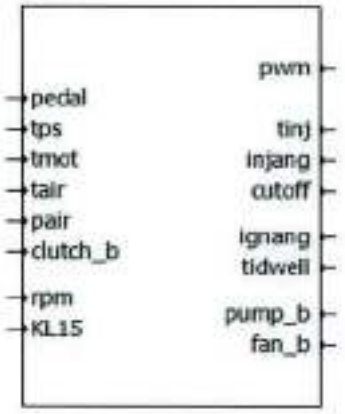
O projeto consiste em gerenciamento de um motor à combustão interna, desta maneira temos como entradas e saídas:

- IN: Posição do pedal (**pedal1**);
- IN: Posição do corpo de borboleta (**tps1**)
- IN: Pressão do ar (**pair**);
- IN: Temperatura do ar (**tair**);
- IN: Temperatura da água do radiador (**tmot**);
- IN: Posição da chave de ignição (**KL15**);
- IN: Sensor de roda fônica*;
- IN: Sensor de fase*;
- IN: Pedal da embreagem (**clutch_b**)
- OUT: Controle de corpo de borboleta eletrônico (**pwm**);
- OUT: Tempo e ângulo de injeção (**tinj** e **injang**);
- OUT: Tempo e ângulo de ignição (**tidwell** e **ignang**);
- OUT: Acionamento da bomba de combustíveis (**pump_b**);
- OUT: Acionamento do ventilador (**fan_b**);

*: sensor de roda fônica e sensor de fase juntos formam a entrada de rotação (**rpm**).

Em todo o desenvolvimento do AppSW foi utilizado um padrão de definição de nome de variáveis para facilitar e interpretação do SW. Este padrão consistem em combinações de prefixos, nome da variável e sufixos que quando combinados representam a função da variável. Mais detalhes no APENDICE D.

Figura 27 – Nível 0 do AppSW



Fonte: o Autor

Tabela 6 – Nível 0 do AppSW

Módulo	Projeto
Entradas	Pedal1, tps1, pair, tmot, tair, rpm e KL15
Saídas	PWM, tidwell, ignangl, tinj, injang, fan_b e pump_b
Funcionalidade	Este bloco gerencia os atuadores necessários para controlar um motor à combustão interna, baseando-se nos sensores lidos, incluindo dentro deste os controles de injeção, ignição, borboleta e relés.

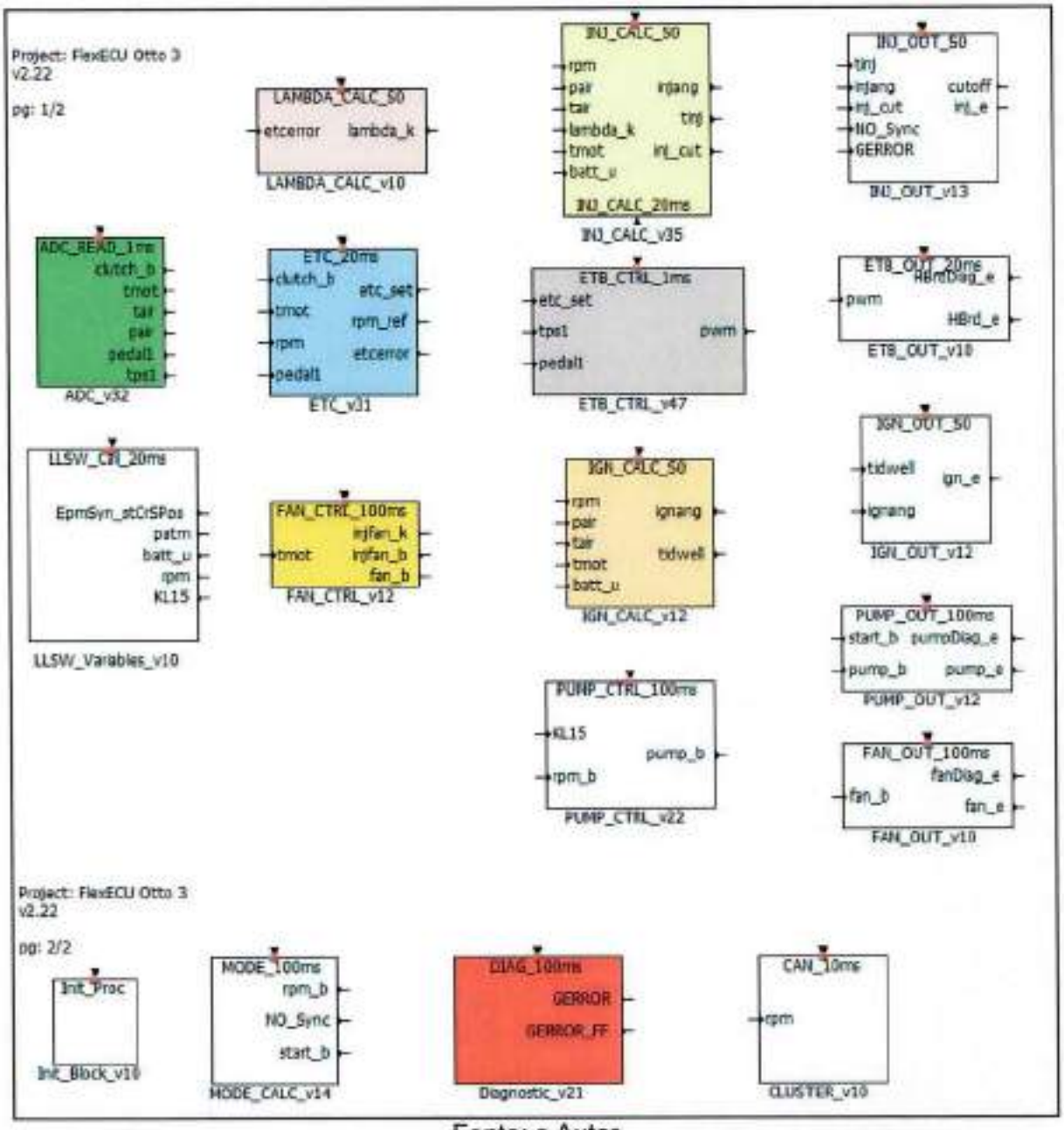
Fonte: o Autor

Analisando a decomposição na Figura 28, é possível notar a ausência de um bloco exclusivo para *safety*. Isso se deve a simplicidade do AppSW como um todo, onde todas as funções de safety foram implementadas dentro de cada bloco considerando uma abordagem nível 1 das estratégias de segurança.

Se este bloco de segurança fosse colocado como apenas um bloco exclusivo, o projeto fugiria da metodologia de *Model-Based Development* uma vez que este bloco dependeria especificamente das demais funções. Se em algum momento de algum projeto seguinte algum dos blocos fosse removido, o AppSW se tornaria não funcional com prováveis falsos alarmes de erros.

Embora em ECUs mais complexas haja estratégias de segurança de níveis 1, 2 e 3, a simplicidade do AppSW deste projeto fez com que fosse optada a inserção de funções de *safety* de nível 1, i.e., dentro de cada bloco.

Figura 28 – Nível 1 do AppSW



Fonte: o Autor

Tabela 7 – Nível 1 do AppSW

Módulo	Projeto Nível 1	Descrição
Blocos	ADC_READ	Lê o valor dos sensores mostrados na Figura 28
	LLSW_CALL	Recebe valores de entradas do LLSW (KL15 e RPM)
	INJ_CALC	Calcula os valores de tempo e ângulo de injeção (também define o *cutoff* de combustível)
	IGN_CALC	Calcula os valores de tempo e ângulo e injeção
	ETB_CTRL	Dado um *setpoint* desejado de borboleta, este bloco busca este ponto através de um controlador PI
	PUMP_CTRL	Define quando deve-se ligar a bomba de combustível
	FAN_CTRL	Define quando deve-se ligar o ventilador
	LAMBDA_CALC	Calcula qual o lambda desejado no momento
	ES_CTRL	Define-se o *setpoint* desejado da borboleta
	INJ_OUT	É um *driver* de injeção, que recebe os valores indicados e aciona o injetor no momento e com o tempo correto.
	IGN_OUT	É um *driver* de ignição, que recebe os valores indicados e aciona a bobina no momento e com o tempo correto.
	ETB_OUT	Aciona a ponte H com o PWM desejado
	PUMP_OUT	*Driver* de acionamento do relé da bomba
	FAN_OUT	*Driver* de acionamento do relé do ventilador do radiador
	MODE	Calcula o estado do motor (marcha lenta, partida e etc.)
	DIAG	Corresponde ao diagnóstico de todos os *drivers* de entrada e saída
	CLUSTER	Envia o valor de rotação do motor para o cluster via CAN
	INIT_BLOCK	Bloco que inicializa algumas variáveis.

Fonte: o Autor

Quanto ao bloco de diagnóstico do sistema, ele é responsável por gravar os erros tanto de HW quanto de SW e gravar um *freeze-frame*, i.e., guarda o valor instantâneo das variáveis de diagnóstico em um vetor no momento do erro do erro, pois, caso ele seja momentâneo ou intermitente, ele não passará despercebido.

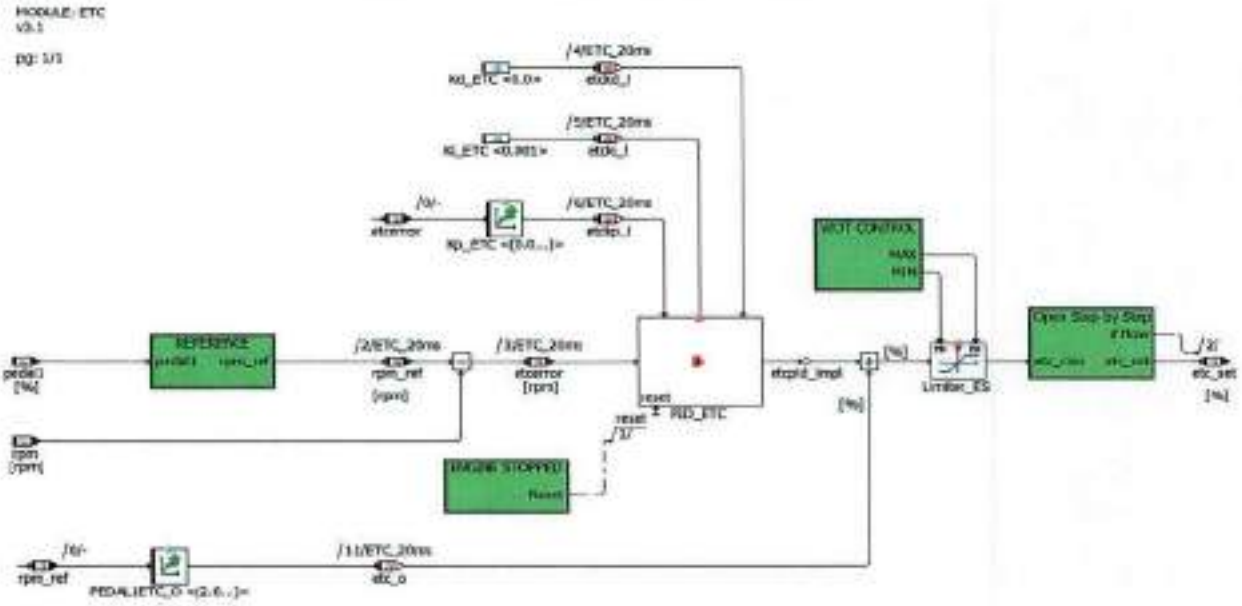
Para a decomposição de nível 2, foram selecionados os blocos do controle propriamente dito para detalhamento.

5.4.1. Controle de ETC

5.4.1.1. Decomposição funcional

O controle de ETC é responsável pela definição do *setpoint* do corpo de borboleta baseado na posição do pedal de acelerador. Nele, diversos parâmetros são calibrados de forma a manipular a resposta do motor em relação à operação do motorista, representada pelo "etc_set", assim, este bloco é um dos principais responsáveis pela dirigibilidade do veículo.

Figura 29 – Diagrama do controle ETC



Fonte: o Autor

Para esta operação, temos a seguinte tabela de entradas e saídas:

Tabela 8 – Variáveis Controle ETC

TIPO	NOME	DESCRIÇÃO
Entrada	pedal1	Valor da posição do pedal em porcentagem após curva de conversão
Entrada	RPM	Velocidade atual da rotação do motor
Entrada	rpmpedal1	Valor de RPM proporcional à posição do pedal do acelerador
Mapa	Kp_ETC	Mapa de conversão do Kp em relação ao erro entre rotação desejada e atual para a posição do ETB
Parâmetro Calibração	Kd_ETC	Coeficiente do termo derivativo do PID que atua no controle do ETC
Parâmetro Calibração	Ki_ETC	Coeficiente do termo integrador do PID que atua no controle do ETC
Saída	etc_set	*Setpoint* de % de abertura do ETB desejado
Subsistema	REFERENCE	Transforma o valor do potenciômetro do pedal do acelerador em um rpm correspondente levando em consideração estratégias de temperatura do motor
Subsistema	ENGINE STOPPED	Identifica quando o motor desligou e dispara um sinal de Reset para as variáveis internas ao PID.
Subsistema	WOT CONTROL	Indica os limites mínimo e máximo da excursão do ETB levando em consideração o rpm atual. Se menor que um certo rpm, limita a ação do WOT, permitindo um *target* máximo menor que a abertura total.
Subsistema	Open Step-by-Step	Limita a variação de % de abertura de cada iteração
Variável Interna	Etcerror	Diferença entre rotação de referência (*setpoint*) e atual
Variável Interna	etc_o	*Offset* somado à saída do PID (estratégia de controle)

Fonte: o Autor

5.4.1.2. Funcionamento

O sinal do pedal (em %) entra no subsistema REFERENCE de onde é obtido o valor de rpm de referência considerando a temperatura do motor. Com este valor, o erro em relação ao rpm atual é calculado e enviado ao PID.

O PID calcula o apenas o incremento necessário no *target* do ETB para atingir o rpm desejado e soma este incremento ao valor de *offset* resultante do mapa PEDAL1ETC_O que indica a posição ideal para o ETB em situações sem carga. Esta estratégia permite um controle mais fino e sutil das acelerações e desacelerações. A compensação de pressão atmosférica que seria relevante para este *offset* não foi considerada neste trabalho uma vez que essa correção não poderia ser calibrada na situação real já que o veículo não pode deixar os limites da Cidade Universitária.

O subsistema *ENGINE STOPPED* serve apenas para fazer com que o PID retorne para as condições iniciais de partida, evitando que um comportamento acumulado anteriormente interfira no controle de partida.

Para evitar que o motor "afogue", é necessário limitar a abertura máxima do ETB. Esta função cabe ao subsistema *WOT CONTROL*, que modifica este limite em duas faixas de rotação. Também cabe a este subsistema limitar o *target* mínimo possível, para evitar que uma condição de estrangulamento seja enviada adiante no sistema.

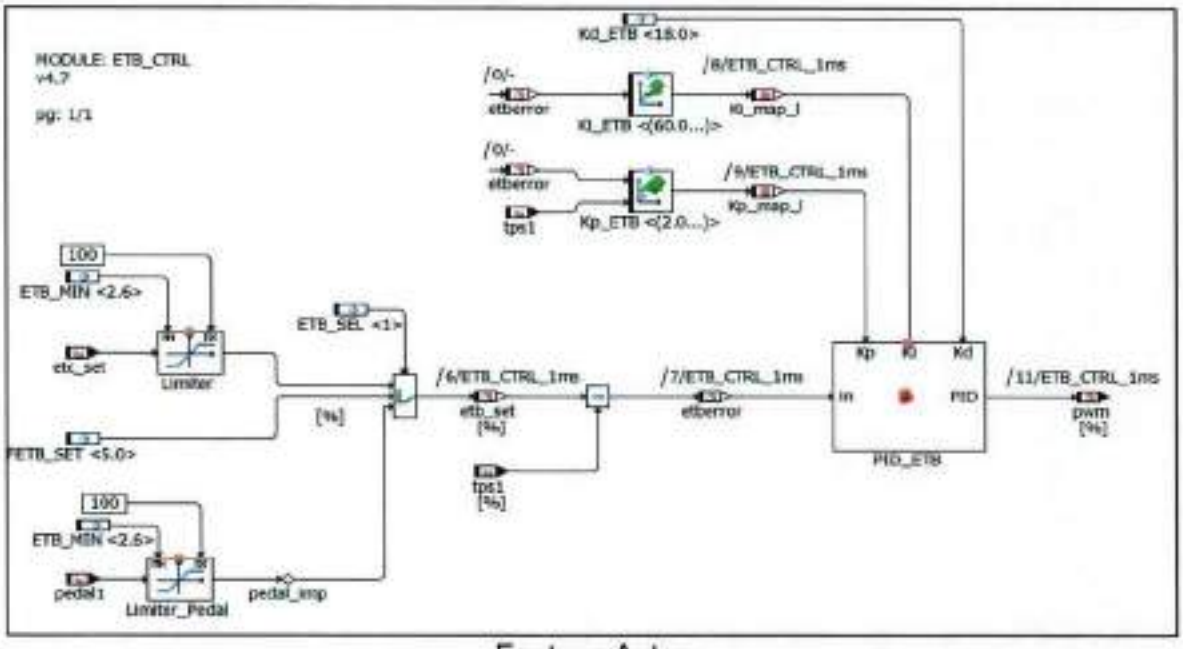
Por fim, o subsistema *Open Step-by-Step* controla a dirigibilidade em relação à resposta do motor à ação do motorista. Nele, uma taxa de variação máxima é definida, de forma a suavizar a abertura do ETB e consequentemente a subida de rpm do motor.

5.4.2. Controle de ETB

5.4.2.1. Decomposição funcional

O controle de ETB basicamente é responsável por transformar a % de abertura estipulada pelo "ETC" (ver detalhes em 0) em um *dutycycle* do PWM que controla o corpo de borboleta.

Figura 30 – Diagrama Controle ETB



Fonte: o Autor

Para esta operação, temos a seguinte tabela de entradas e saídas:

Tabela 9 – Variáveis Controle ETB

TIPO	NOME	DESCRIÇÃO
Entrada	etc_set	*Target* de % de abertura do ETB desejado pelo ETC
Entrada	pedal1	Valor da posição do pedal em porcentagem após curva de conversão
Entrada	tps1	Posição atual do ETB
Mapa	Ki_ETB	Mapa de conversão do Ki em relação ao erro da posição do ETB
Mapa	Kp_ETB	Mapa de conversão do Kp em relação ao erro da posição do ETB e à posição atual do ETB
Parâmetro Calibração	ETB_MIN	Mínima abertura que o ETB pode atingir
Parâmetro Calibração	FETB_SET	Permite ao INCA forçar um *target* de %de abertura de ETB
Parâmetro Calibração	ETB_SEL	Seleciona a origem do *target* da posição do ETB entre ETC, diretamente do pedal ou forçado pelo INCA
Parâmetro Calibração	Kd_ETB	Coeficiente do termo derivativo do PID que atua no controle do ETB
Saída	pwm	*Dutycycle* do PWM que fará a mudança na posição do ETB desejada pelo controle
Variável Interna	etb_set	Posição desejada enviada ao controle após seleção da fonte do *target* do ETB
Variável Interna	Ki_map_I	Coeficiente do termo integrador do PID que atua no controle do ETB
Variável Interna	Kp_mapI	Coeficiente do termo proporcional do PID que atua no controle do ETB
Variável Interna	etberror	Diferença entre posição desejada e atual do ETB

Fonte: o Autor

5.4.2.2. Funcionamento

Para cada valor de "ETB_SEL", uma das fontes é selecionada. Para operação normal, esta fonte deve ser o "etc_set", que corresponderá à dirigibilidade calibrada, para o levantamento dos mapas de avanço e mistura, a fonte utilizada foi o "FETB_SET", pois permite um controle mais preciso e fixo da carga e, para permitir uma flexibilidade em diversos testes, a fonte pode ser diretamente do pedal do acelerador, sem controle algum no caminho, permitindo impor qualquer abertura em qualquer situação.

Para as fontes dinâmicas, uma limitação era imposta aos valores imputados, de forma a evitar *targets* impossíveis como, por exemplo, o estrangulamento do ETB.

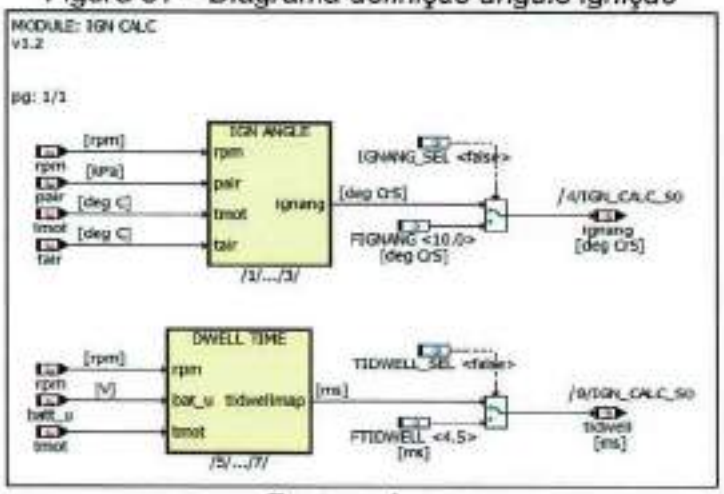
Após esta fase, o *target* final é obtido e o erro calculado. Este erro indica o quanto o ETB deve abrir ou fechar para atingir o ponto desejado. Este erro entra em um PID, com Kp e Ki variáveis em função do erro e do movimento do ETB, abertura ou fechamento, que resultam no PWM necessário para realizar o movimento desejado.

5.4.3. Cálculo do ângulo de ignição

5.4.3.1. Decomposição funcional

A função "IGN_CALC" analisa os valores das medições dos sensores e calcula o ângulo de ignição ideal para aquele ponto de operação bem como o tempo de carregamento da bobina.

Figura 31 – Diagrama definição ângulo ignição



Fonte: o Autor

Para esta operação, temos a seguinte tabela de entradas e saídas:

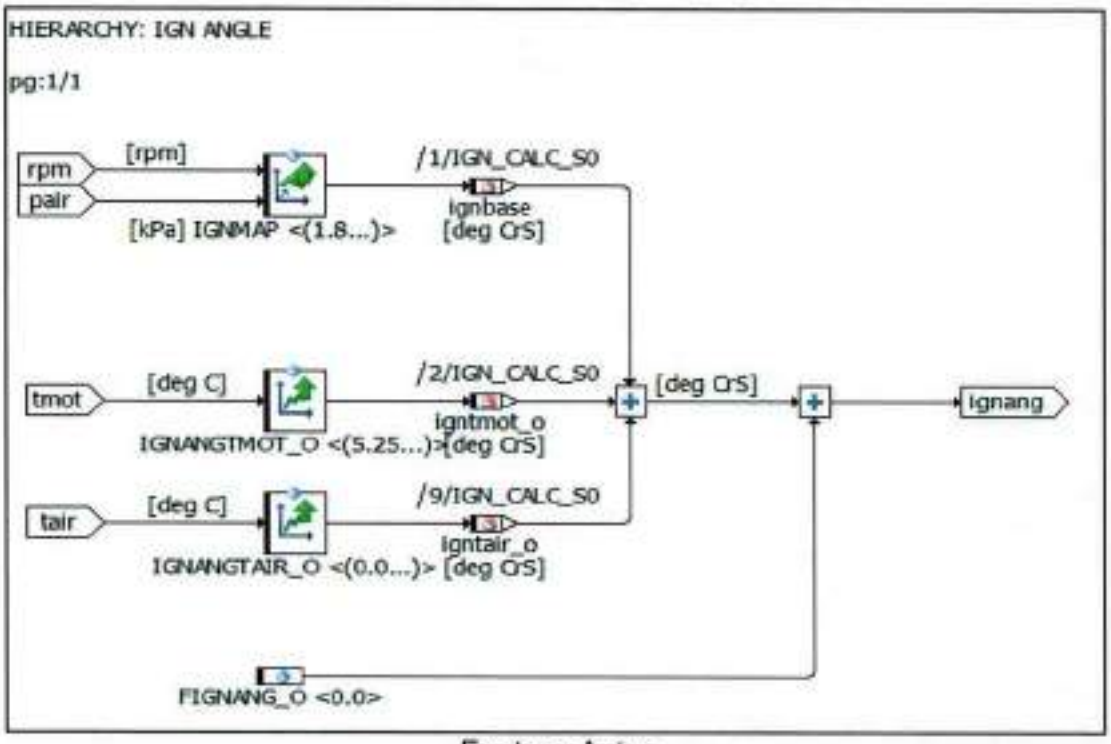
Tabela 10 – Variáveis definição ângulo ignição

TIPO	NOME	DESCRIÇÃO
Entrada	RPM	Rotação atual do motor
Entrada	pair	Pressão do ar aspirado em kPa
Entrada	tmot	Temperatura do motor em °C
Entrada	tair	Temperatura do ar aspirado em °C
Entrada	batt_u	Valor da tensão da bateria
Parâmetro Calibração	IGNANG_SEL	Permite alternar entre ângulo de ignição calculado e forçado
Parâmetro Calibração	FIGNANG	Permite ao INCA forçar um ângulo de ignição
Parâmetro Calibração	TIDWELL_SEL	Permite alternar entre tempo de carregamento da bobina calculado e forçado
Parâmetro Calibração	FTIDWELL	Permite ao INCA forçar um tempo de carregamento da bobina
Saída	ignang	Ângulo de ignição enviado ao *driver* de ignição
Saída	tidwell	Tempo de carregamento da bobina enviado ao *driver* de ignição
Subsistema	IGN ANGLE	Realiza o cálculo do ângulo de ignição
Subsistema	DWELL TIME	Tempo de carregamento da bobina

Fonte: o Autor

Em detalhe, temos o subsistema de cálculo do ângulo de ignição.

Figura 32 – Cálculo ângulo ignição



Fonte: o Autor

Para este subsistema, foram adicionadas as seguintes variáveis:

Tabela 11 – Variáveis ângulo de ignição

TIPO	NOME	DESCRIÇÃO
Mapa	IGNMAP	Recebe rpm e pressão do ar e devolve o valor base de ângulo de ignição
Mapa	IGNANGTMOT_O	Dada uma temperatura do motor, devolve um offset para o ângulo base
Mapa	IGNANGTAIR_O	Dada uma temperatura do ar, devolve um *offset* para o ângulo base
Parâmetro Calibração	FIGNANG_O	Permite ao INCA forçar um *offset* ao ângulo de ignição calculado
Saída	Ignang	Resultado do cálculo do ângulo de ignição

Fonte: o Autor

5.4.3.2. Funcionamento

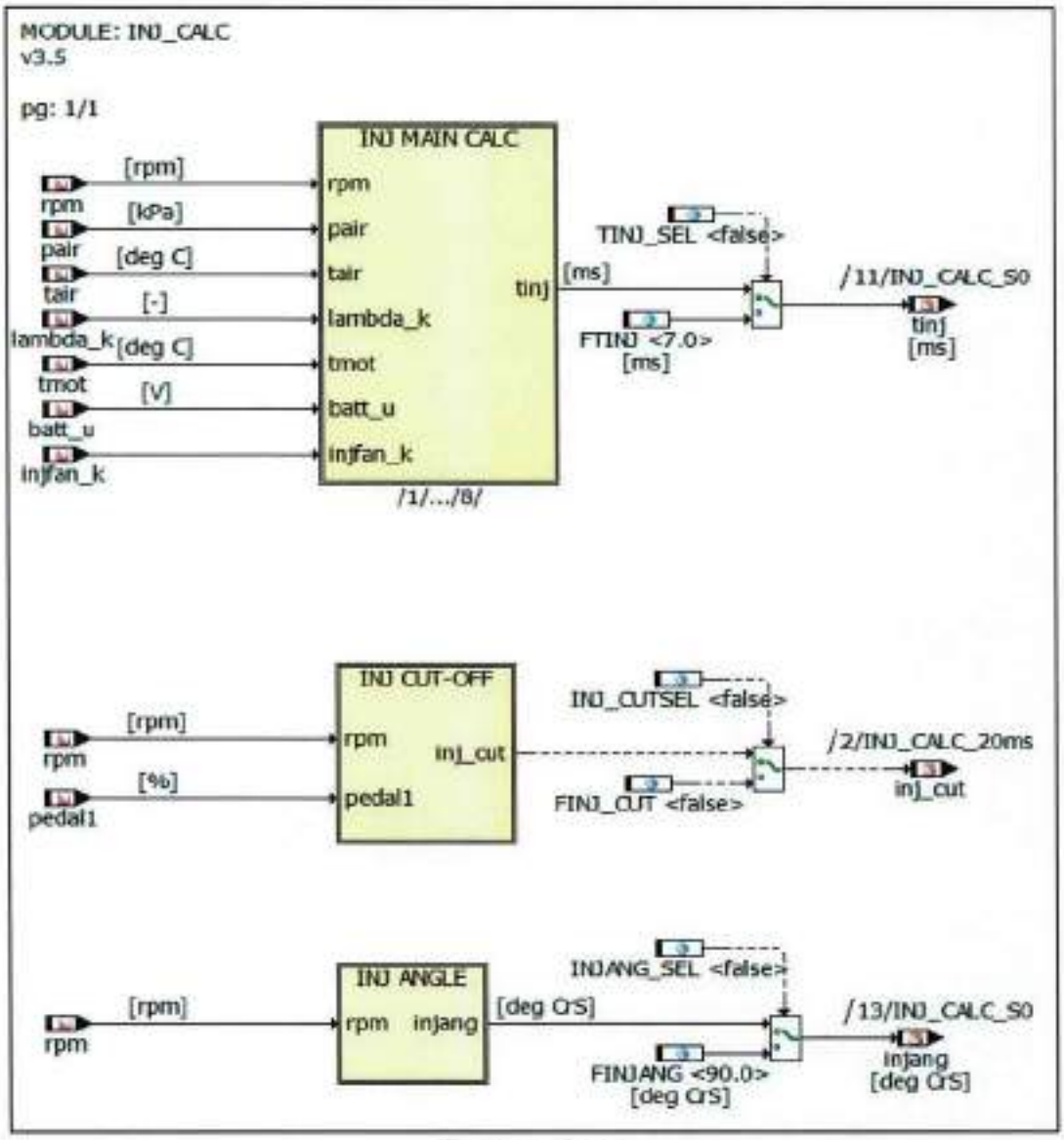
O tempo de carregamento da bobina e o ângulo de ignição são selecionados entre cálculo e valor forçado pelo INCA e são enviados ao *driver* de ignição. Os valores calculados são fornecidos pelos respectivos subsistemas.

O cálculo do ângulo de ignição é simples. Dado um ponto de operação, um ângulo base é fornecido pelo "IGNMAP" e, de acordo com as condições do motor e do ambiente, correções são somadas a este ângulo base.

5.4.4. Cálculo do tempo de injeção

5.4.4.1. Decomposição funcional

Figura 33 – Definição tempo de injeção



Fonte: o Autor

Para esta operação, temos a seguinte tabela de entradas e saídas:

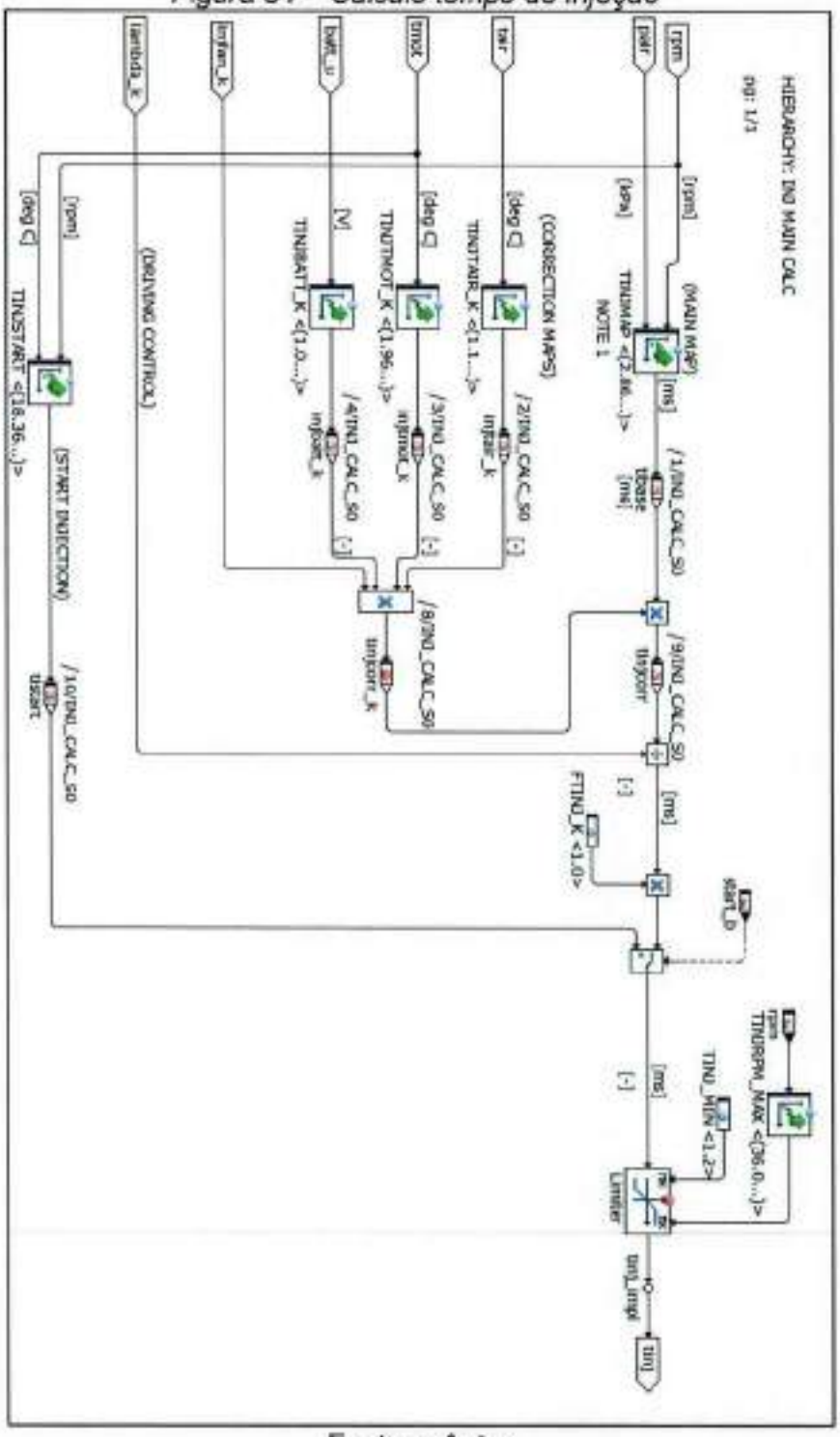
Tabela 12 – Variáveis definição tempo de injeção

TIPO	NOME	DESCRIÇÃO
Entrada	RPM	Rotação atual do motor
Entrada	Pair	Pressão do ar aspirado em kPa
Entrada	Tmot	Temperatura do motor em °C
Entrada	Tair	Temperatura do ar aspirado em °C
Entrada	batt_u	Valor da tensão da bateria
Entrada	lambda_k	Fator de enriquecimento da mistura para acelerações
Entrada	injfan_k	Fator de enriquecimento da mistura para compensar a demanda de torque adicionada pela ventoinha
Entrada	pedal1	Valor da posição do pedal em porcentagem após curva de conversão
Parâmetro Calibração	TINJ_SEL	Permite alternar entre tempo de injeção calculado e forçado
Parâmetro Calibração	FTINJ	Permite ao INCA forçar um tempo de injeção
Parâmetro Calibração	INJ_CUTSEL	Permite alternar entre a utilização da estratégia ou forçar um *cutoff*
Parâmetro Calibração	FINJ_CUT	Permite ao INCA forçar um *cutoff*
Saída	Tinj	Tempo de injeção enviado ao *driver* de injeção
Saída	inj_cut	Sinal de *cutoff* enviado ao *driver* de injeção
Subsistema	INJ MAIN CALC	Realiza o cálculo do tempo de injeção
Subsistema	INJ CUT-OFF	Executa a estratégia de *cutoff*

Fonte: o Autor

Em detalhe, temos o subsistema de cálculo do tempo de injeção:

Figura 34 – Cálculo tempo de injeção



Fonte: o Autor

Para este subsistema, foram adicionadas as seguintes variáveis:

Tabela 13 – Variáveis cálculo tempo de injeção

TIPO	NOME	DESCRIÇÃO
Mapa	TINJMAP	Recebe rpm e pressão do ar e devolve o valor base de tempo de injeção
Mapa	TINJTMOT_K	Dada uma temperatura do motor, devolve um fator a ser multiplicado no tempo base
Mapa	TINJTAIR_K	Dada uma temperatura do ar, devolve um fator a ser multiplicado no tempo base
Mapa	TINJBATT_K	Dada uma tensão de bateria, devolve um fator a ser multiplicado no tempo base
Mapa	TIINJSTART	Mapa referente à operação especial de startup, contendo tempos de injeção necessários para partida
Mapa	TINJRPM_MAX	Dado um rpm, limita o tempo máximo de injeção que pode ser enviado ao *driver*
Parâmetro Calibração	FTINJ_K	Permite ao INCA forçar um fator a ser multiplicado no tempo corrigido de injeção
Parâmetro Calibração	TINJ_MIN	Limita o mínimo tempo de injeção que pode ser enviado ao *driver*
Saída	Tinj	Resultado do cálculo do tempo de injeção
Variável Interna	tinjcorr_k	Condensa todos os fatores de correção
Variável Interna	Tinjcorr	Armazena o tempo de injeção corrigido
Variável Interna	start_b	Variável que seleciona entre modo de operação normal e *startup*.

Fonte: o Autor

5.4.4.2. Funcionamento

Esta função fornece ao *driver* de injeção o sinal de *cutoff*, que faz com que o sistema corte a injeção de combustível, calcula o tempo e ângulo de injeção referente às leituras dos sensores.

O cálculo do tempo de injeção ocorre em quatro fases.

- Seleção de modo de operação

  Se o motor está parado, ao tentar da partida a variável start_b será acionada até que, pela estratégia de partida, o modo é alternado para operação normal;

- Cálculo do tempo base

  Dadas as leituras de rpm e pressão do ar, o mapa principal devolve um tempo de injeção base, que considera as premissas feitas na calibração dos mesmos;

- Correção dos fatores

  Para que o sistema leve em consideração pontos de operação diferentes aos utilizados no ensaio, uma série de correções é necessária. Nesta fase, as diferenças causadas por diferenças na temperatura do ar, temperatura do motor e tensão da bateria são levadas em conta em forma de fatores multiplicativos.

- Fator de aceleração

  Para que o motor responda melhor e mais rapidamente à uma aceleração, dependendo do tamanho do erro, um enriquecimento da mistura ar-combustível é feita, aumentando o tempo já corrigido.

  No caso de acelerações bruscas, algumas estratégias se aliam à essa de forma a suavizar a aceleração sem comprometer a resposta, principalmente reduzindo o pico de lambda maior que 1.

  - Enriquecimento prévio à abertura da borboleta

    Pela aceleração gerar um aumento rápido da massa de ar, o sistema antecipa esta ação e já enriquece a mistura antes de permitir esta quantidade de ar adicional entre na câmara, diminuindo o efeito de empobrecimento da mistura.
  - Abertura da borboleta *Step-by-step* (ver detalhes em 5.4.1.2)

Por fim, o sistema garante que o tempo desejado esteja dentro dos limites corretos de operação, onde o máximo varia em função do rpm, de forma a evitar injeção de combustível de maneira ineficaz.

5.5. Medição e Calibração

Após gerar as principais funções para o funcionamento do sistema, e.g. sincronismo, ignição e injeção de combustível, um ponto de partida era necessário para os mapas de ignição e de injeção.

5.5.1. Mapa básico de injeção

Para a criação deste mapa, era necessário saber a quantidade de ar aspirada pelo motor em todas as condições de operação para, por meio do cálculo estequiométrico da combustão de gasolina, calcular a quantidade de combustível necessária.

Após uma análise dos sensores disponíveis no carro, uma estratégia de cálculo era necessária devida a ausência de um sensor MAF. Para utilização de um sensor MAP, a seguinte estratégia de cálculo foi definida.

Partindo da equação geral para um gás ideal (I), algumas premissas físicas restringem a medição a apenas pressão do ar, resultando, matematicamente, na massa de ar aspirada (II):

- 100% de eficiência volumétrica
- Temperatura constante; o cálculo considera temperatura constante de 40°C, sendo corrigido posteriormente pela função de correção de tempo de injeção (ver detalhes em 5.4.4.2)

I) $$P.V.VE = \frac{m}{M}.R.T$$

II) $$m_{air} = \frac{P_{air} * V_{cyl} * VE * M_{air}}{R * T_{air}}$$

Obtida a quantidade de ar, o próximo passo é o cálculo da quantidade proporcional de combustível. Considerando uma mistura ar-combustível perfeita, i.e., lambda igual a 1, a massa necessária é dada por (III):

III) $$m_{fuel} = \frac{m_{air}}{13.3}$$

O tempo de injeção foi calculado considerando o fluxo do injetor de combustível original de fábrica equivalente a 2,5 g/s. (IV)

IV) $$t_{inj} = m_{fuel} * \frac{1}{flux}$$

Combinadas todas as fases do cálculo, a equação de tempo de injeção em função de pressão do ar (V) foi obtida utilizando os dados de técnicos do motor (ver detalhes em 4.1)

V) $$t_{inj}(p_{air}) = \frac{\frac{P_{air} * V_{cyl} * M_{air}}{R * T_{air}}}{13,3} * \frac{1}{flux}$$

Como a eficiência volumétrica (VE) de 100% não existe para este motor, o mapa precisa ser preparado para incluir este dado. A principal causa da menor eficiência volumétrica é o tempo de admissão do ar em cada ciclo, que diminui proporcionalmente ao aumento do rpm, assim a VE é uma função da rotação:

VI) $$t_{inj}(p_{air}) = \frac{\frac{P_{air} * f(rpm) * M_{air}}{R * T_{air}}}{13{,}3} * \frac{1}{flux}$$

No entanto esta função não se torna simples na ausência de um sensor MAF, portanto neste projeto a VE foi corrigida empiricamente no dinamômetro de chassis em todos os pontos de operação (detalhes em 5.5.3.1).

Uma alteração na resolução da conversão do sensor de pressão do ar foi necessária devida à sensibilidade do controle na partida onde, imediatamente antes e no início do transiente, a pressão ficava em torno de 80 e 90 kpa, tornando necessária uma maior resolução neste intervalo.

Tabela 14 – Mapa de tempos de injeção

Injection Time [ms] - Calculated Map														
	rpm [1/min]													
	500	1000	1500	2000	2500	3000	3500	4000	4500	5000	5500	6000	6500	7000
pair [kPa] 20	2,6742	2,6742	2,6742	2,6742	2,6742	2,6742	2,6742	2,6742	2,6742	2,6742	2,6742	2,6742	2,6742	2,6742
30	4,011	4,011	4,011	4,011	4,011	4,011	4,011	4,011	4,011	4,011	4,011	4,011	4,011	4,011
40	5,348	5,348	5,348	5,348	5,348	5,348	5,348	5,348	5,348	5,348	5,348	5,348	5,348	5,348
50	6,686	6,686	6,686	6,686	6,686	6,686	6,686	6,686	6,686	6,686	6,686	6,686	6,686	6,686
60	8,023	8,023	8,023	8,023	8,023	8,023	8,023	8,023	8,023	8,023	8,023	8,023	8,023	8,023
70	9,360	9,360	9,360	9,360	9,360	9,360	9,360	9,360	9,360	9,360	9,360	9,360	9,360	9,360
80	10,697	10,697	10,697	10,697	10,697	10,697	10,697	10,697	10,697	10,697	10,697	10,697	10,697	10,697
90	12,034	12,034	12,034	12,034	12,034	12,034	12,034	12,034	12,034	12,034	12,034	12,034	12,034	12,034
100	13,371	13,371	13,371	13,371	13,371	13,371	13,371	13,371	13,371	13,371	13,371	13,371	13,371	13,371
110	14,708	14,708	14,708	14,708	14,708	14,708	14,708	14,708	14,708	14,708	14,708	14,708	14,708	14,708
120	16,045	16,045	16,045	16,045	16,045	16,045	16,045	16,045	16,045	16,045	16,045	16,045	16,045	16,045

Fonte: o Autor

5.5.2. Mapa básico de ignição

Para o mapa básico de ignição, foi escolhida uma abordagem cautelosa. Completando o mapa primariamente com 3.75 graus de avanço, ajustes proporcionais foram feitos. Um maior avanço foi aplicado proporcionalmente ao aumento do rpm e diminuição da pressão.

Essa simplificação do mapa de ignição (Tabela 15) se deve ao fato de ele não ser crucial para o funcionamento ou não do motor, sendo mais importante para o desempenho, emissões, etc.

Tabela 15 – Mapa básico tempos de injeção

Ignition Angle [deg CrS] - Calculated Map														
	rpm [1/min]													
	500	1000	1500	2000	2500	3000	3500	4000	4500	5000	5500	6000	6500	7000
pair [kPa] 20	3,75	3,75	9,75	21,00	21,00	24,75	25,50	26,25	21,75	22,50	22,50	22,50	22,50	22,50
30	3,75	3,75	9,75	20,25	20,25	24,75	25,50	26,25	21,75	22,50	22,50	22,50	22,50	22,50
40	3,75	3,75	9,75	20,25	19,50	24,75	25,50	24,00	19,50	22,50	22,50	22,50	22,50	22,50
50	3,75	3,75	9,75	19,50	19,50	22,50	24,75	23,25	18,75	22,50	22,50	22,50	22,50	22,50
60	3,75	3,75	9,75	19,50	18,75	21,00	24,75	22,50	18,00	22,50	22,50	22,50	22,50	22,50
70	3,75	3,75	9,75	19,50	18,75	18,00	22,50	22,50	18,00	21,75	21,75	21,75	21,75	21,75
80	3,75	3,75	9,75	19,50	18,75	18,00	20,25	20,25	17,25	19,50	19,50	19,50	19,50	19,50
90	3,75	3,75	9,75	15,00	18,00	18,00	19,50	19,50	17,25	19,50	19,50	19,50	19,50	19,50
100	3,75	3,75	9,75	15,00	17,25	17,25	18,75	18,75	17,25	18,75	18,75	18,75	18,75	18,75
110	3,75	3,75	9,75	15,00	17,25	17,25	18,75	18,75	17,25	18,75	18,75	18,75	18,75	18,75

Fonte: o Autor

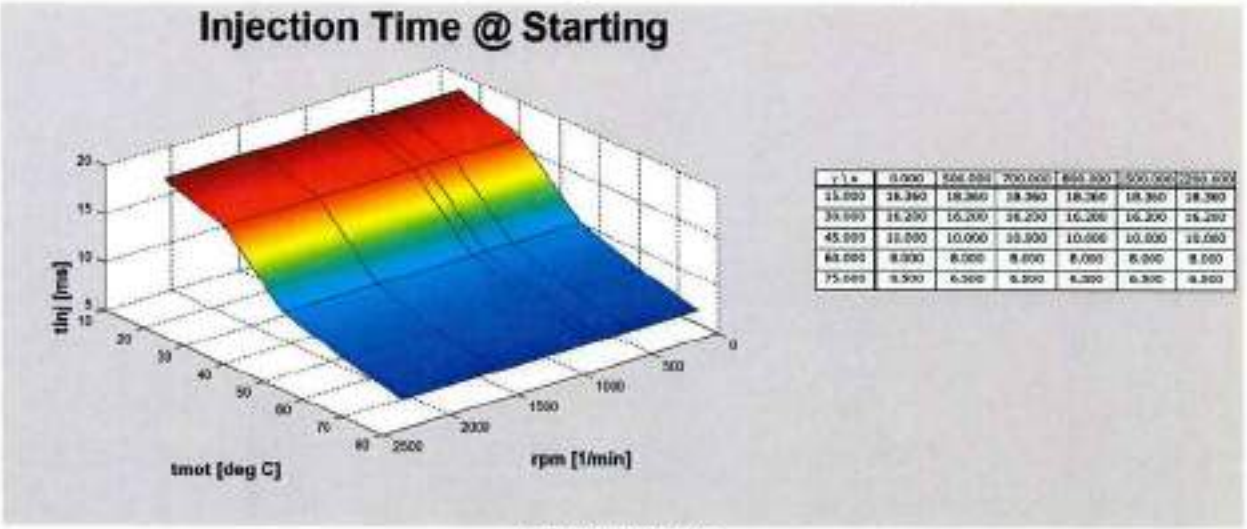
5.5.3. Refinamento dos mapas

Como o funcionamento normal do motor difere muito da partida, para o refinamento o espectro de velocidade do motor foi dividido em dois blocos, *startup/idle* indo de 0 a 1500rpm e operação normal a partir de 1500rpm.

Startup

O primeiro passo foi descobrir o tempo mínimo de injeção para *startup* com temperatura do motor igual a 90°C mantendo um tempo de injeção fixo durante a partida e incrementando-o a cada iteração. Como a quantidade de combustível necessária para partida é maior do que a necessária para operação normal uma correção diferente das já existentes foi necessária, portanto o ajuste do startup foi feito juntamente com a calibração da função de partida, que faz com que o mapa resultante (Figura 35) entre em operação desde a partida até o fim de um ciclo pré-estabelecido.

Figura 35 – Mapa de injeção de combustível na partida



Fonte: o Autor

O principal parâmetro desta função é quando interromper sua atuação, sendo definido por um determinado tempo após o motor ultrapassar uma determinada rotação.

Idle

Para a operação em *idle*, um método de calibração diferente, em relação ao da operação normal, foi necessária. Para que fosse possível a manutenção da rotação do motor abaixo de 1000 rpm, foi necessária obter uma mistura rica nestas situações (ver Figura 36) aliada às estratégias para grandes acelerações, de forma que o motor não "morresse" nas saídas em que o condutor agisse suavemente.

O fato da ausência de um controle de torque e uma função de predição de demanda de torque não permitiu um ajuste fino de compensação do funcionamento da direção hidráulica, o que obrigou uma marcha lenta ligeiramente mais alta que garantisse uma reserva de torque suficiente ao *idle* do motor e ao funcionamento da direção hidráulica em manobras que exigem um esforço maior, como por exemplo, as de estacionamento.

Operação normal

Devido à falta de um dinamômetro de bancada, a calibração dos mapas foi feita utilizando um dinamômetro de chassis.

Para um ajuste homogêneo, um procedimento foi estabelecido para tentar diminuir os erros e desvios relativos às múltiplas variáveis de um dinamômetro de chassis.

- Utilizar sempre terceira marcha;
- Temperatura do motor entre 85 e 95 graus Celsius;

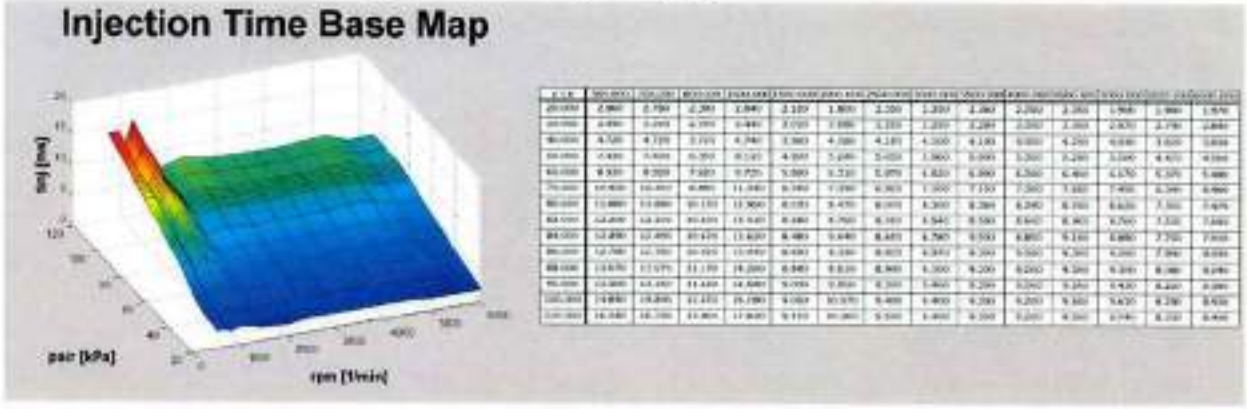
- Temperatura do ar entre 40 e 50 graus Celsius;
- Bypass do acelerador e grau de abertura de ETB definido pelo INCA
- Rpm controlado dinamicamente pelo dinamômetro

Para os testes, foram utilizados *steps* de 500rpm e 5% de ETB. Como não é possivel induzir uma pressão de ar controlada, a variação da mesma se dará pela maior abertura do ETB e maior rotação

5.5.3.1. Injeção

Para calibrar o mapa de injeção, em cada teste, o ponto estável era atingido e o tempo de injeção era ajustado até lambda ficar igual a 1. Neste ponto, as medições de pressão do ar e tempo de injeção formam um par e são adicionados ao mapa.

Figura 36 – Tempo de injeção base final



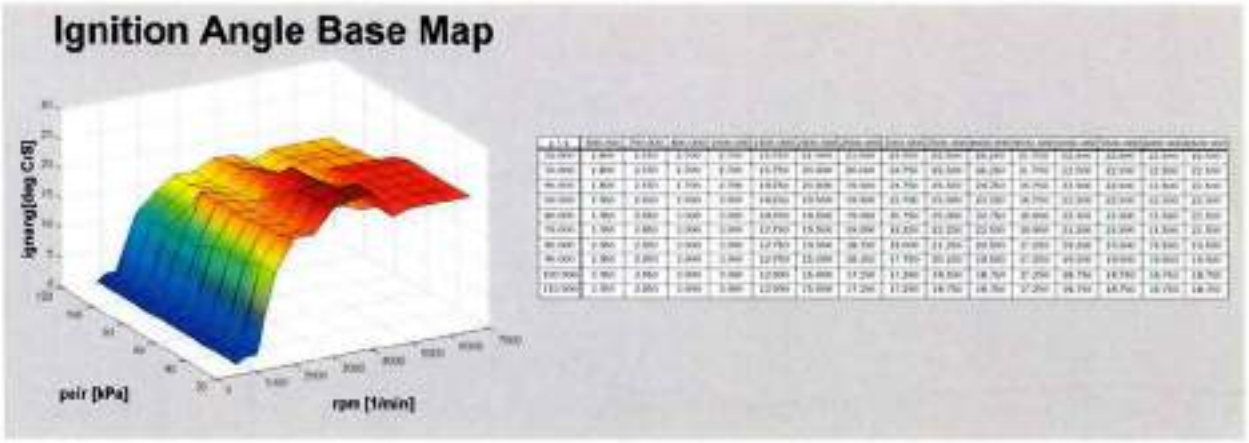
Fonte: o Autor

5.5.3.2. Ignição

Para a ignição, o dinamômetro não só mantinha dinamicamente a rotação como também fornecia a medição do torque instantâneo na roda. Somada a esta informação, também foi necessário um aparato para detecção de detonação.

O ajuste ocorria da seguinte forma. Após estabilizado o ponto de operação e mantido lambda igual a 1, avanço era dado até que o torque máximo fosse obtido ou a detonação era identificada, o que ocorresse primeiro (processo para encontro do MBT).

Figura 37 – Ângulo de ignição base final



Fonte: o Autor

5.5.3.3. Detecção de detonação

Este sistema consistia de um fone de ouvido conectado diretamente na saída do sensor de detonação. A detecção era possível, pois o funcionamento normal do motor produz uma vibração específica que corresponde a um zumbido constante. Já a ocorrência da detonação resulta em um chiado intermitente e bastante agudo, claramente diferente da operação normal.

As dificuldades encontradas nessa etapa foram relacionadas ao ruído inerente do próprio funcionamento do motor e ausência de isolamento acústico entre operação do dinamômetro e o rolo.

Para baixas cargas baixas rotações o fenômeno de *knock* e máximo torque eram difíceis de serem detectados e por isso uma abordagem conservadora foi adotada com relação ao avanço para evitar danos ao motor. No caso de altas cargas e rotações, o ruído relacionado ao funcionamento do motor no sinal de *knock* era bem intenso o que dificultava e detecção da detonação, por isso, foi inserido alguns filtros digitais para tratar o sinal e destacar as frequências de ocorrência de detonação (entre 5 kHz e 15 kHz). Nas outras regiões de rotação e carga o *knock* era suficientemente audível com o sistema utilizado.

5.5.3.4. Pontos não aquisitados

Dada a dificuldade do controle da pressão do ar, vários pontos de pressão de ar não ocorreram durante os testes. Analisando os dados, foi calculada a variação média entre pontos subsequentes, tanto para aumento de carga quanto para aumento de rpm, e estes fatores foram aplicados ao mapa para extrapolação dos dados.

5.5.4. Calibração do Sistema

5.5.4.1. Teste em laboratório

Ao redor do Sistema, vários parâmetros de calibração foram dispostos de uma forma que o sistema pudesse responder a um grande número de possibilidades. Para estes ajustes gerais, muitas horas foram gastas para tornar a operação segura para seguir para a rua.

Como principais pontos e parâmetros, podemos citar

- Funções de segurança

  As funções de segurança foram massivamente testadas e calibradas. Para uma operação segura, várias análises qualitativas e quantitativas foram feitas durante operações incorretas propositais e falhas simuladas para o sistema atingir parâmetros aceitáveis de tempo de resposta, assertividade da estratégia de reação.

- Resposta do ETB

  Para um funcionamento suave do motor, o comportamento da abertura do ETB foi calibrado por muitas horas. Dado um *setpoint* o ETB deve atingi-lo o mais rápido possível, com o mínimo de *overshoot*, *undershoot* e oscilação, pois estes fatores influenciam no comportamento do motor.

- Resposta do ETC

  Como foi utilizado um controle de rotação, o sinal do pedal precisava ser tratado de uma forma a passar ao usuário uma sensação mais próxima ao usual controle de torque. Como isso é crucial para o comportamento dinâmico do carro, essa calibração foi definida como mandatória para o *green light* para o *road test*. E.g. uma abertura muito rápida fazia o carro "pular" e, uma muito lenta, diminuía muito a velocidade de retomada.

5.5.4.2. *Road Test*

Após considerado seguro o funcionamento do carro, o primeiro *road test* foi marcado e o carro preparado. Para simplificar os ajustes e análises durante o teste, alguns dispositivos foram instalados dentro do carro para permitir, tanto ao passageiro quanto ao motorista, monitorar e modificar as variáveis além da utilização de um botão de emergência que desliga a ECU caso algum problema seja identificado.

Nesta fase, a meta era calibrar o sistema de modo a melhorar as características dinâmicas do funcionamento, o que se resumiu, basicamente, a calibrar os diversos parâmetros dos controles de ETC e ETB.

- Saídas suaves e retomadas
    - Terreno plano

Para suavizar as saídas em parada completa em terreno plano dois ajustes foram fundamentais, velocidade de resposta do ETB e ajuste da curva de conversão do pedal eletrônico do acelerador.

Um ETB muito ágil fazia com que o carro oscilasse demais nas saídas lentas pois, o sistema ao enxergar uma demanda de rotação, iterava os *steps* de abertura previamente calibrados de forma muito rápida, não dando tempo para o motor "encher", fazendo o carro dar diversos trancos e oscilar bastante o rpm até atingir uma velocidade maior.

O ajuste do pedal se deve mais ao controle de rotação. Em terrenos planos, pouco torque é necessário para uma saída em primeira marcha, mas com uma curva linear de pedal, um simples toque no pedal já fazia o motor fornecer muito mais torque do que o necessário, dado que buscava uma rotação alta demais muito rápido.

Para minimizar este efeito, uma curva não linear foi implementada, suavizando a variação da meta de rpm em um primeiro estágio do pedal e aumentando progressivamente até o fim.

    - Aclive

O ajuste neste caso se dava pelo fato da abertura ter ficado lenta demais após o ajuste da saída em terreno plano. A solução veio através da modificação do controle de ETC, onde passou-se a limitar a %de abertura e o máximo de abertura (através do "etc_set" – funções *WOT Control* e *Open Step-by-Step*) em determinadas rotações e pressionamento do *clutch*. Unindo uma boa calibração destas funções à um ajuste na curva de conversão do pedal (conversão de posição de pedal para referência de rpm), foi possível calibrar de forma satisfatória para todos os tipos de carga em aclives.

- Compensação da direção hidráulica em manobras

Um ajuste no mapa de injeção se fez necessário devido a constantes falhas do motor em manobras de 3 pontos e balizas, devido a direção hidráulica somar uma carga extra (não esperada) ao motor. Como o *idle* foi calibrado para uma rotação próxima

de 800rpm, esse aumento de carga unido ao aumento de demanda dada pela aceleração para manobra fazia com que o motor apagasse.

Para diminuir este efeito, o *idle* passou a ser mantido com mistura ligeiramente rica, para ter uma reserva de torque para estes casos.

- *Cutoff*

  A estratégia de *cutoff* foi importante em duas situações, prevenir um *overshoot* de rotação na troca de marcha e aumentar o freio motor e economizar combustível em declives acentuados.

  O problema estava em estas situações possuírem estratégias opostas, o que inviabilizava um ajuste satisfatório. A solução veio na adição do sinal do pedal de embreagem (ver detalhes em 6.3), permitindo um atraso no início no *cutoff* em situações que não se estivesse trocando de marcha.

- *Throttle Kick down*

  Para melhorar a resposta do carro quando o motorista "pisa fundo", o final da curva do pedal foi alterada para buscar um rpm impossível, fazendo com que o motor entregasse todo o torque possível naquele momento, ignorando calibrações de conforto e suavidade, dando um perfil mais esportivo a uma condução mais agressiva.

6. Resultados e Discussões

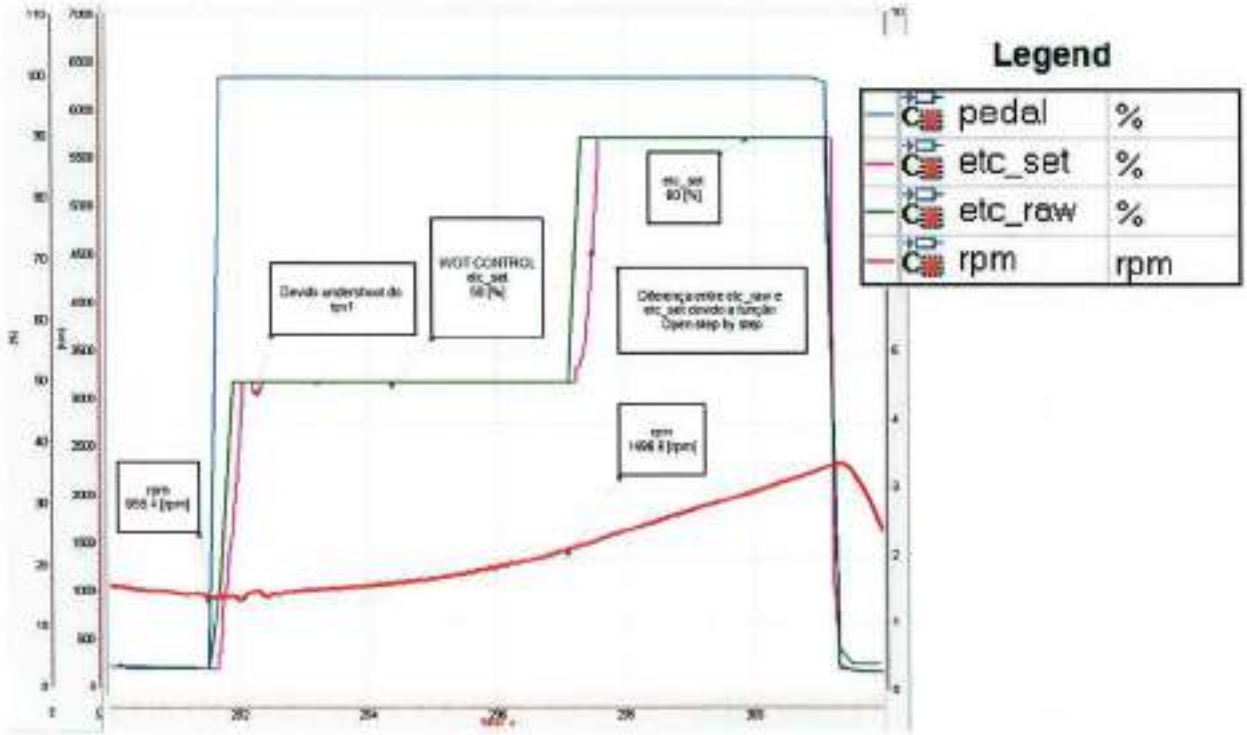
Após todo o desenvolvimento e testes, uma série de resultados qualitativos e quantitativos foram colhidos para que uma avaliação do resultado pudesse ser feita. Em relação aos quantitativos, pode-se citar tomadas de dados em situações específicas como aceleração em WOT e quanto aos quantitativos, o convite de pessoas com diferentes níveis de conhecimento técnico para testar e dar um *feedback* com suas impressões.

A seguir, as tomadas quantitativas mais significativas para o bom funcionamento do sistema foram selecionadas para uma análise detalhada.

6.1. Função ETC

Conforme explicado no item 0 este bloco consiste em basicamente definir o *setpoint* do corpo de borboleta. Neste projeto, um controle de rotação foi aplicado, no entanto, foi inserida uma série de blocos auxiliares para tornar a dirigibilidade do veículo algo confortável e com poucos trancos. Na Figura 38 é possível observar estes blocos auxiliares atuando em uma situação de WOT.

Figura 38 – Funcionamento ETC em uma situação de WOT



Fonte: o Autor

Na medição apresentada, é possivel observar o funcionamento de duas funções separadamente:

- "*Open Step-by-Step*": responsável pela diferença do "etc_raw" (curva verde) e "etc_set" (curva rosa) essa diferença permite um enriquecimento prévio da mistura antes da abertura do ETB e evita uma abertura muito brusca da mesma, o que pode causar trancos no veiculo.
- "*WOT CONTROL*": responsável pelo patamar de 50% mostrado no gráfico, pois mesmo com uma demanda de pedal de WOT (curva azul), como o motor se encontra em uma rotação baixa (curva vermelho - 955.4 rpm) uma abertura repentina de ETB causaria um tranco. Por isso, a borboleta abre no máximo 50% até atingir uma rotação maior que 1500 e só então volta a abrir até atingir 90% (não foi utilizado o valor de 100% para evitar que a válvula se chocasse com o fim do curso, além de não haver diferença na admissão com ETB em 90% e 100%).

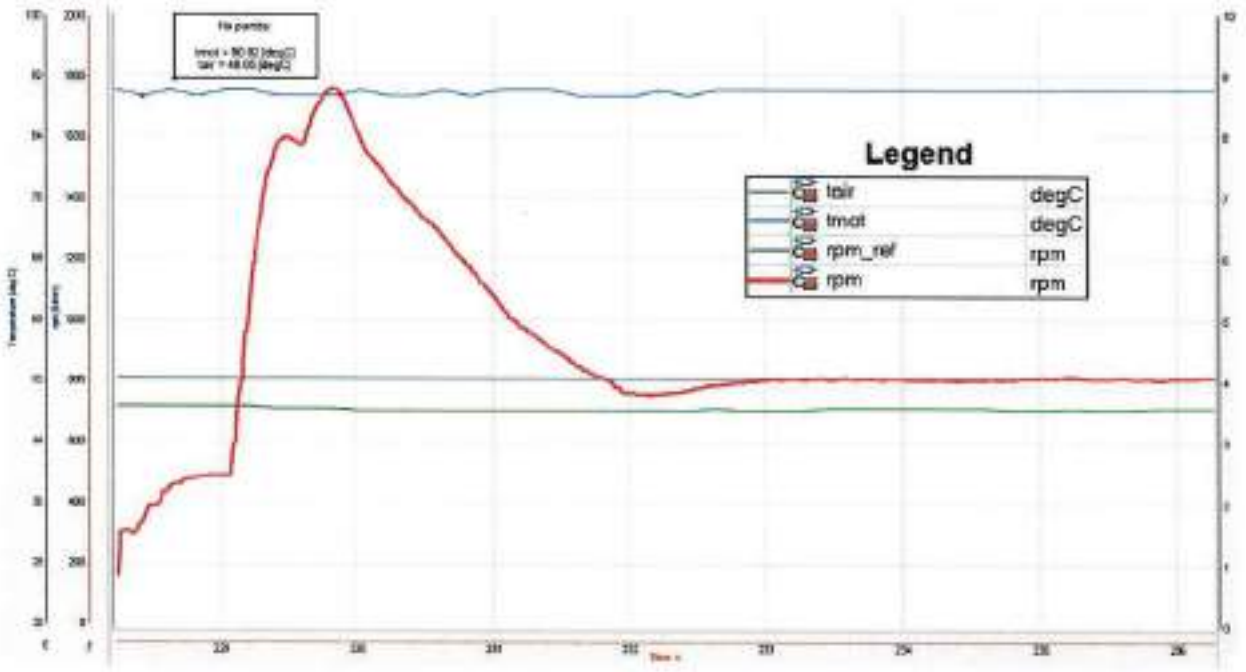
6.2. Partida

O foco do projeto não foi o desenvolvimento da partida, por isso foi desenvolvido e calibrado um AppSW suficiente para partir e estabilizar a marcha lenta em diversas condições de temperatura do motor e ambiente. Assim, não foram aplicadas grandes exigências com relação à curva de lançamento do motor onde em muitas situações o resultado de partir e estabilizar a marcha lenta eram suficientes.

Na Figura 39 mostra-se uma partida em uma situação de motor quente, i.e., temperatura do motor (curva azul) em 90°C e temperatura do ar de admissão (curva verde) em 48°C. A rotação desejada é a curva azul escura e a rotação atual curva vermelha.

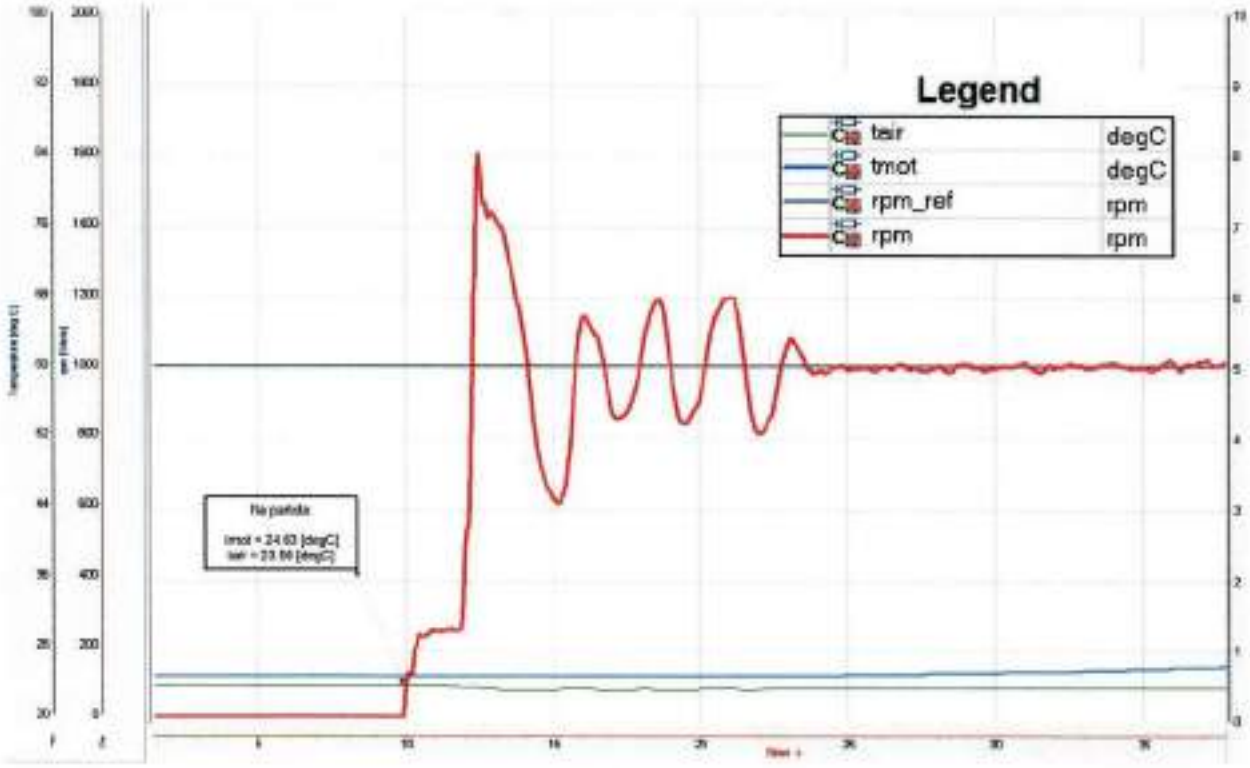
Já na Figura 40, é mostrada uma partida em uma situação de motor frio e desligado a mais de 24h, i.e., temperatura do motor (curva azul) em 24.6°C e temperatura do ar de admissão (curva verde) em 23.6°C. Observa-se que apesar de algumas oscilações da rotação atual (curva vermelha) em torno da rotação desejada (curva azul escura) de 1000 rpm, a marcha lenta foi estabilizada o que para o escopo do projeto é um resultado positivo.

Figura 39 – Partida com motor quente



Fonte: o Autor

Figura 40 – Partida com motor frio e desligado a mais de 24h

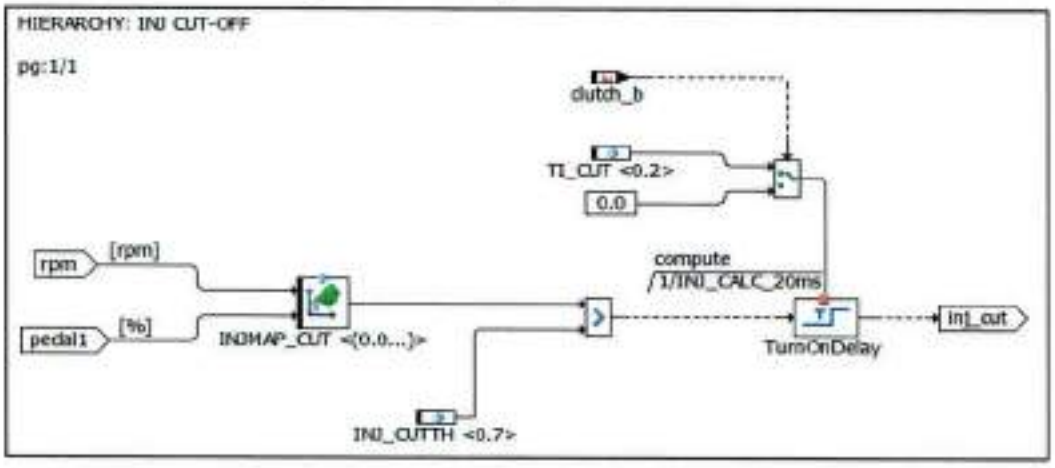


Fonte: o Autor

6.3. Controle de *overshoot* de rotação na troca de marcha

Para evitar o *overshoot* de rotação na troca de marcha, foi inserido uma estratégia no módulo "INJ_CALC" dentro da função "INJ CUT-OFF". Esta função por si só define os momentos de *cutoff* através de um mapa de calibração, incluindo momentos de declives e troca de marcha (como pode ser observado na Figura 41).

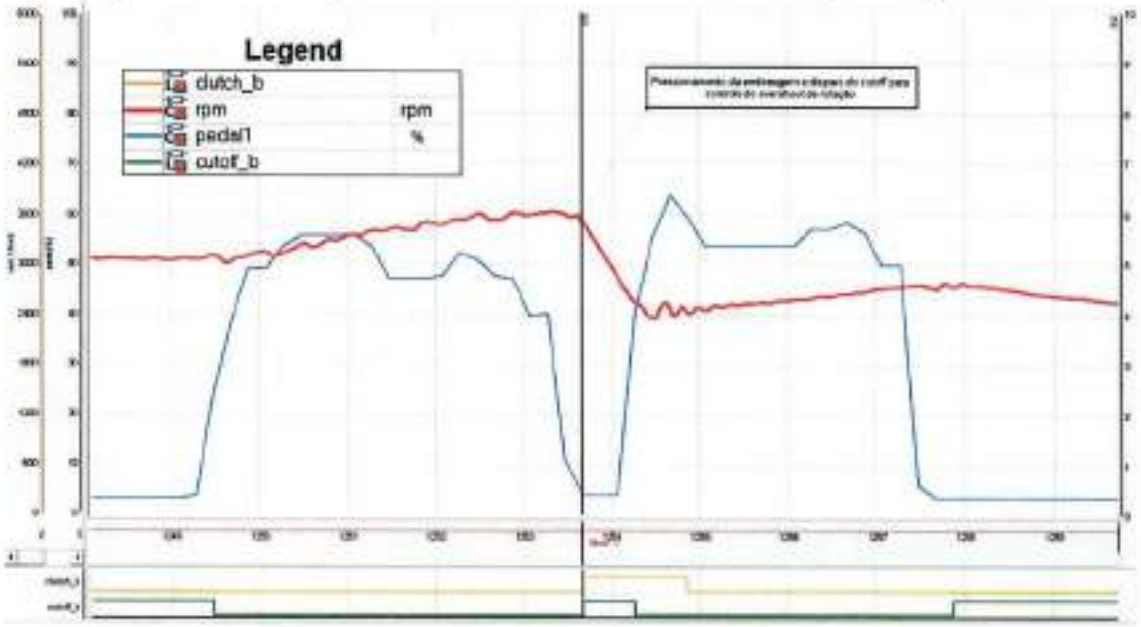
Figura 41 – Função "INJ CUT-OFF"



Fonte: o Autor

Nas trocas de marcha, ao contrário das outras situações, o cutoff precisa atuar imediatamente. Isto pode ser observado na Figura 42, onde o acelerador (curva azul) foi para 0% e o clutch (curva laranja) foi pressionado, assim o cutoff (curva verde) foi acionado controlando a subida de rotação (curva vermelha).

Figura 42 – Cutoff para controle de overshoot em situação de gear shift



Fonte: o Autor

6.4. Curvas de Potência e Torque

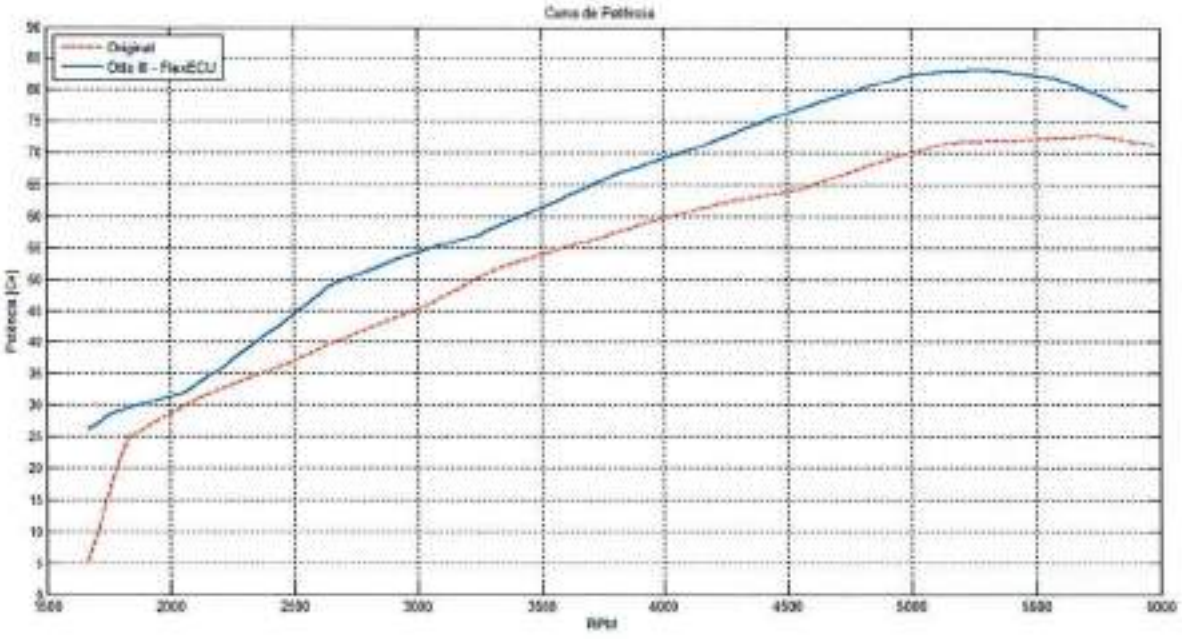
Abaixo encontram-se as curvas de potência e torque obtidas em ensaios no dinamômetro de chassis. Ambas as curvas evidenciam dois ensaios, um com a ECU original e outro com a FlexECU rodando o projeto Otto III. Os parâmetros utilizados nos testes foram:

- Gasolina
- 3ª marcha
- Espectro de RPM: 1500 – 6000

Como o veículo ficou vários anos com uma manutenção insuficiente antes de ser alocado para este projeto, as curvas evidenciam os resultados obtidos sem nenhuma correção, visto que, corrigir a pressão atmosférica, que resultaria em números 10% maiores para a altitude do local do ensaio por exemplo, deixa de fazer sentido quando se tem outros diversos fatores influenciando o resultado.

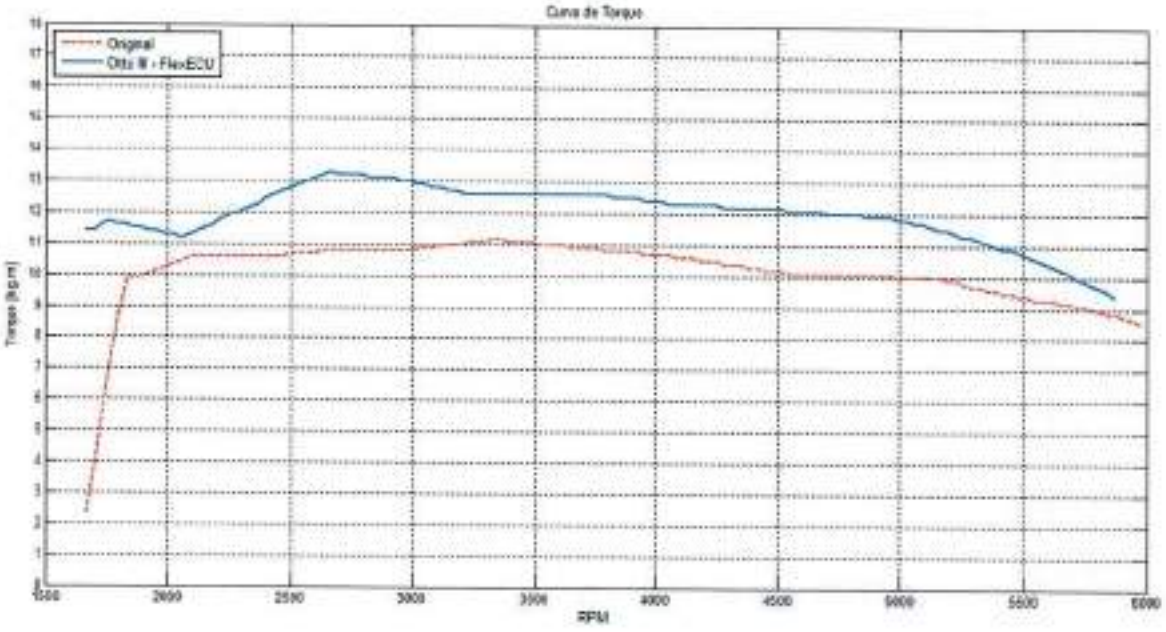
Portanto, as curvas comparam a desempenho dos dois sistemas sob as mesmas condições, embora este projeto não leve em consideração níveis de emissões. Esta comparação serve apenas para avaliar de uma forma numérica e simplificada o desempenho do veículo sob o comando do sistema deste projeto.

Figura 43 – Curva de Potência



Fonte: o Autor

Figura 44 – Curva de Torque



Fonte: o Autor

Tabela 16 – Picos de potência e torque obtidos nos ensaios

	Otto III by FlexECU	**Original**
Potência	83,3 kgf.m @ 5288rpm	72,9 kgf.m @ 5755rpm
Torque	13,3 kgf.m @ 2667rpm	11,2 kgf.m @ 3359rpm

Fonte: o Autor

7. Conclusões

Analisando os dois principais objetivos deste projeto, realizar o desenvolvimento de um AppSW seguindo o máximo possível da metodologia automotiva e criar um sistema com o código aberto e acessível à equipe, de forma que melhorias e novas implementações possam ser feitas futuramente de forma modular e organizada, pode-se chegar à conclusão de dever cumprido.

Um dos grandes empecilhos em pesquisas mais detalhadas por parte do grupo de eletrônica automotiva jaz no fato das ECUs comerciais serem fechadas, não permitindo alterações específicas, pesquisas de novas estratégias de controle e nem uma profunda compreensão do sistema. Empecilho este que o projeto Otto III visou e conseguiu solucionar criando assim um novo horizonte de possibilidades e novas pesquisas.

Como aprendizado para a dupla executora deste projeto, pode-se destacar o que foi decorrente da experiência de efetivamente desenvolver um software de aplicação nos moldes da metodologia e os conhecimentos adquiridos sobre a parte mecânica dos motores.

Estudando a teoria pode-se ter apenas uma ideia dos ganhos e das dificuldades em seguir o *V-model*, o que, em contrapartida, um período de "mão na massa" permite um entendimento mais profundo e exato dela. Porém, uma graduação em engenharia eletrônica não fornece todos os elementos suficientes para a criação de um controle para algo mecânico, então um estudo sobre os detalhes do ciclo Otto foi necessário e trouxe bastante frutos.

Para o grupo e para a universidade, é deixado como legado um sistema modular e aberto, de forma a facilitar futuras pesquisas. Como recomendação para próximos projetos, pode-se destacar criação de novos sistemas e melhorias como:

- Controle de lambda em malha fechada;
- Detecção de *knock* e controle ativo do sistema;
- Desenvolvimento de um gerenciamento para bicombustíveis;
- Controle de torque;
- Calibração seguindo as normas de emissões brasileiras.

8. Referências

SCHÄUFFELE, J.; ZURAWKA, T. (2003). ***Automotive Software Engineering: Principles, Processes, Methods, and Tools***. Tradução do alemão de Roger Carey. Warrendale, PA USA, 2005. 385 p.

PEREIRA, B. C. F ***Unidade de Gerenciamento Eletrônico de um Motor Volkswagen 2.0L: Projeto Otto II.*** Dissertação (Graduação) – Escola Politécnica da Universidade de São Paulo, 2013.

ETAS, Inc., FlexECU-G1 – **Gasoline Low Level Software Interface Definition Description**, V1.1 – Dezembro 2011.

BRUNETTI, F. ***Motores de combustão interna,*** Volumes 1 e 2, 3 ed. São Paulo: Edgar Blücher, 2012.

ETAS. *Site*. Disponível em: http://www.etas.com/en/. Acesso em: abril/2015.

Flyer FlexECU – Site Disponível em: http://www.etas.com/download-center-files/ products _flexECU/FlexECU_en_2015-04-13.pdf. Acesso em: abril/2015.

Ficha Técnica Volkswagen Gol G5 1.6L TOTALFLEX – Site Disponível em: http://estadodeminas.vrum.com.br/fichatecnica/Volkswagen/GOL/2009/005190-0. Acesso em abril/2015.

ES581 – Site Disponível em: http://www.etas.com/de/products/es581.php. Acesso em abril/2015.

ES63x – Site Disponível em: http://www.etas.com/de/products/es63x.php. Acesso em abril/2015.

Sonda Lambda Wideband – Site Disponível em: http://www.aim-store.com/en/ Accessorys/Temperature/Exhaust-gas/Lambda-probe-LSU-49.html. Acesso em abril/2015.

RTA-OSEK – Site Disponível em: http://www.etas.com/en/products/rta_osek.php. Acesso em abril/2015.

OSEK – Site Disponível em: http://www.osek-vdx.org/. Acesso em abril/2015.

Manual ASCET – Site Disponível em: http://www.etas.com/dlc_tmp/a82d10c458cb8eefeeb2 713f0c0d7025_1445788688/ASCET-RP_V6.3_UsersGuide.pdf. Acesso em abril/2015.

Bypass – Site Disponível em: http://www.elektroniknet.de/automotive/tools/artikel/30563. Acesso em abril/2015.

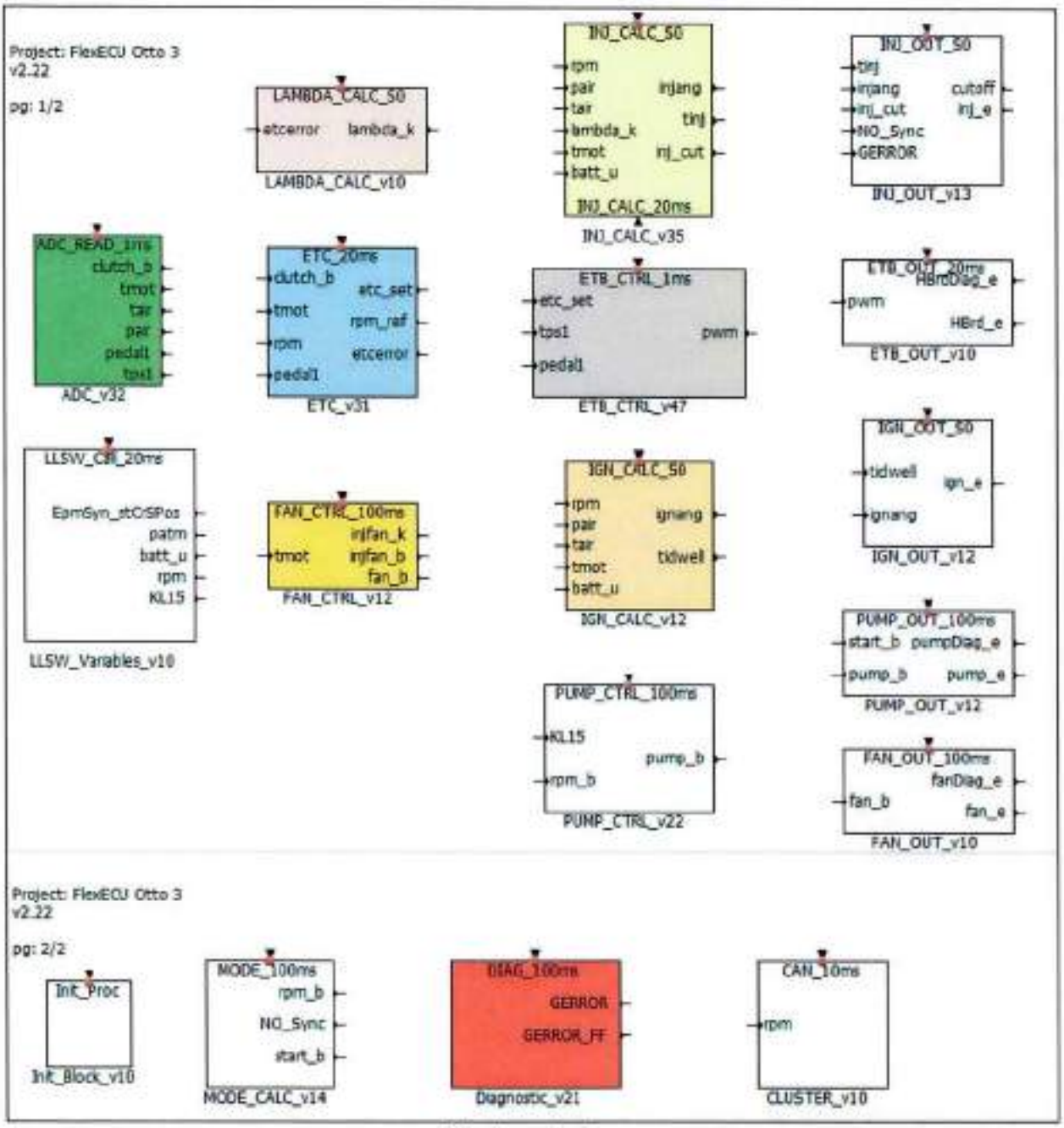
Bypass 2 – Site Disponível em: http://www.elektroniknet.de/automotive/tools/artikel/121095. Acesso em abril/2015

On-target Bypass – Site Disponível em: http://www.etas.com/data/RealTimes_2012/ rt_2012_2_36_en.pdf. Acesso em abril/2015.

Apêndices

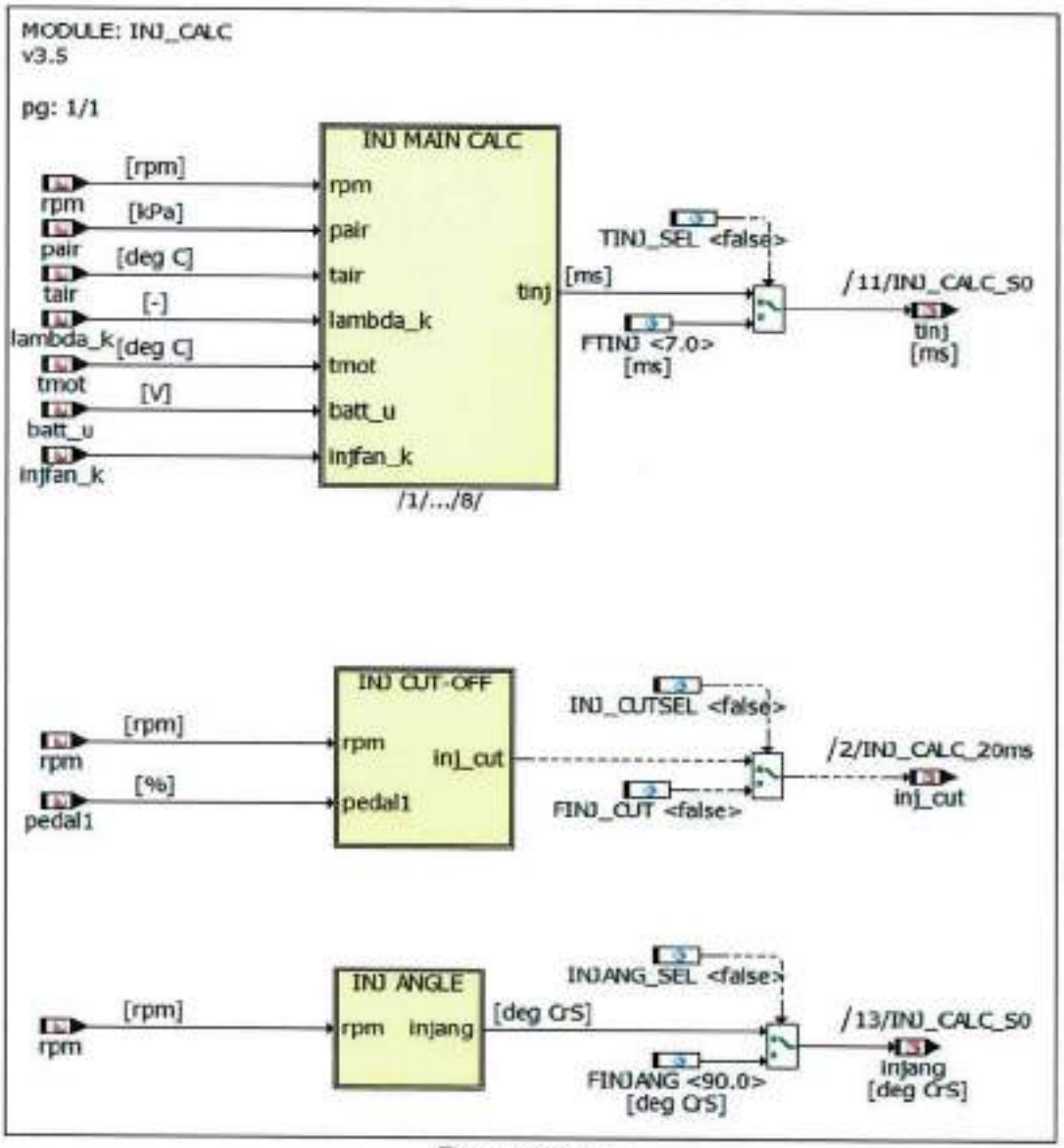
APÊNDICE A – Principais funções do AppSW

Figura 45 – Nível 1 do AppSW



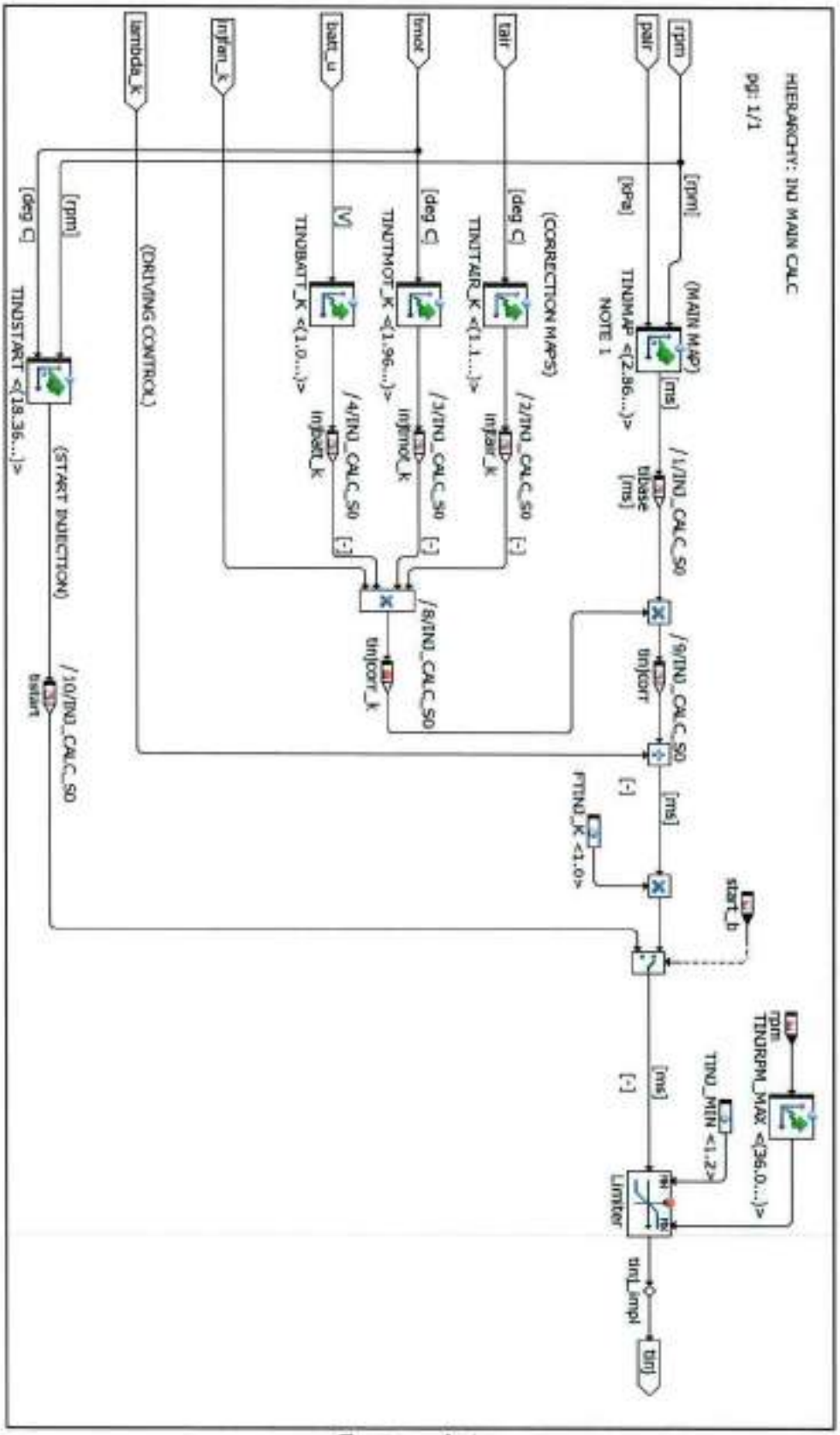
Fonte: o Autor

Figura 46 – Módulo "INJ_CALC"



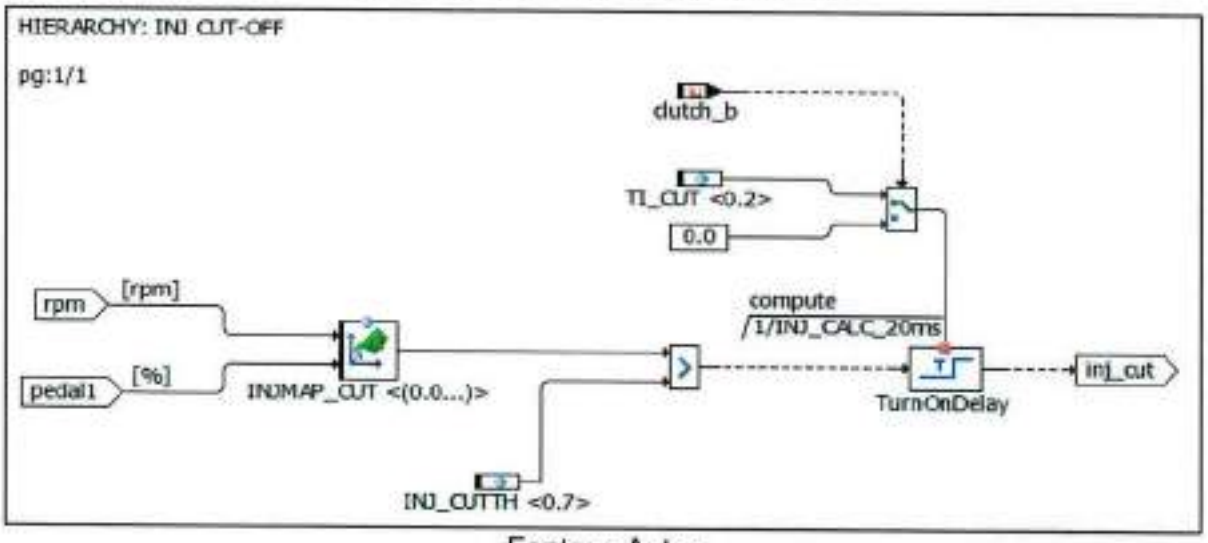
Fonte: o Autor

Figura 47 – Subsistema "INJ MAIN CALC"



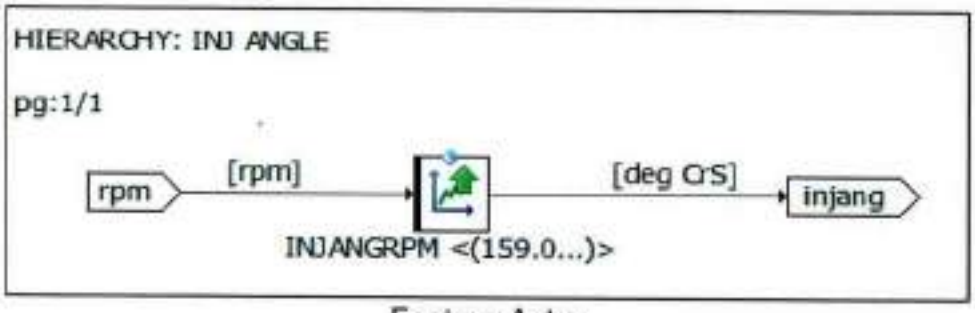
Fonte: o Autor

Figura 48 – Subsistema INJ CUT-OFF



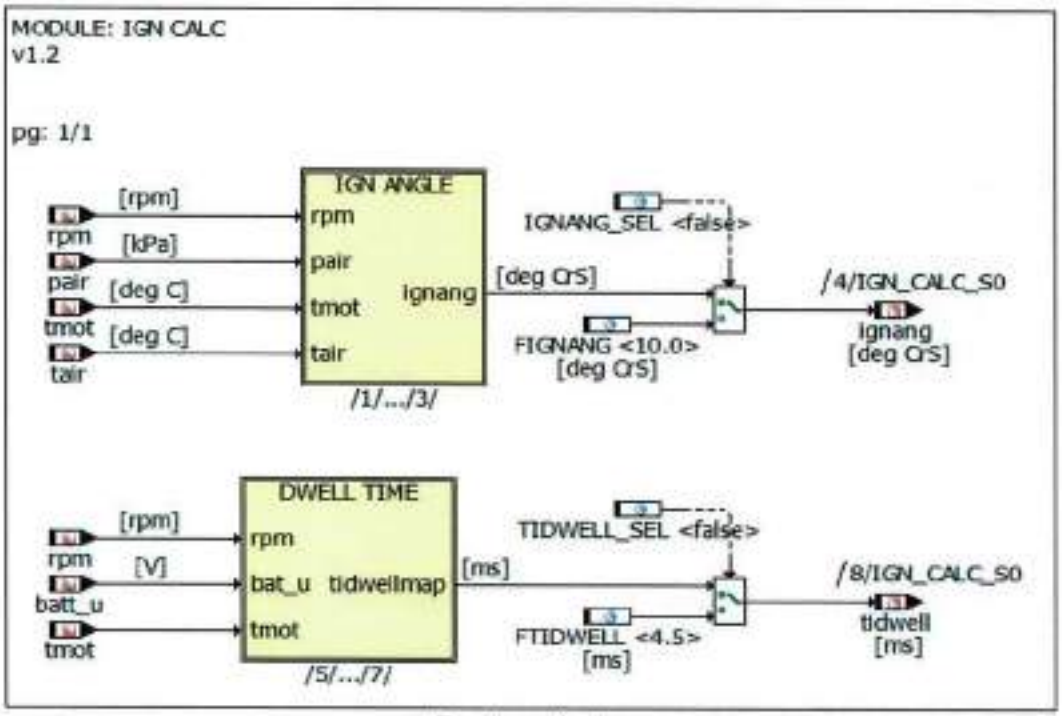
Fonte: o Autor

Figura 49 – Subsistema "INJ ANGLE"



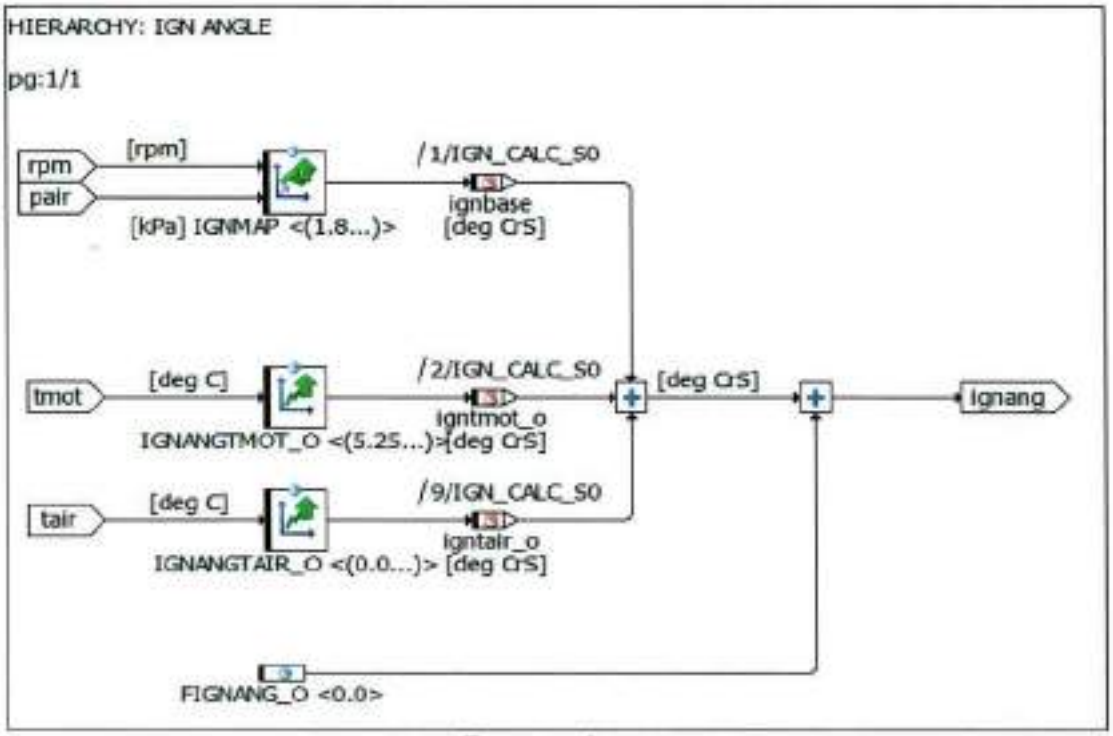
Fonte: o Autor

Figura 50 – Módulo "IGN_CALC"



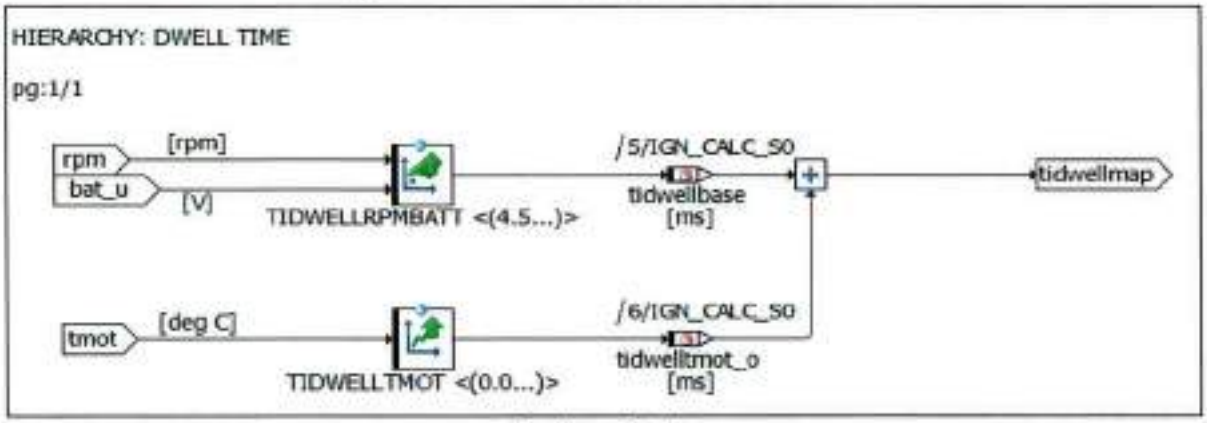
Fonte: o Autor

Figura 51 – Subsistema "IGN_ANG"



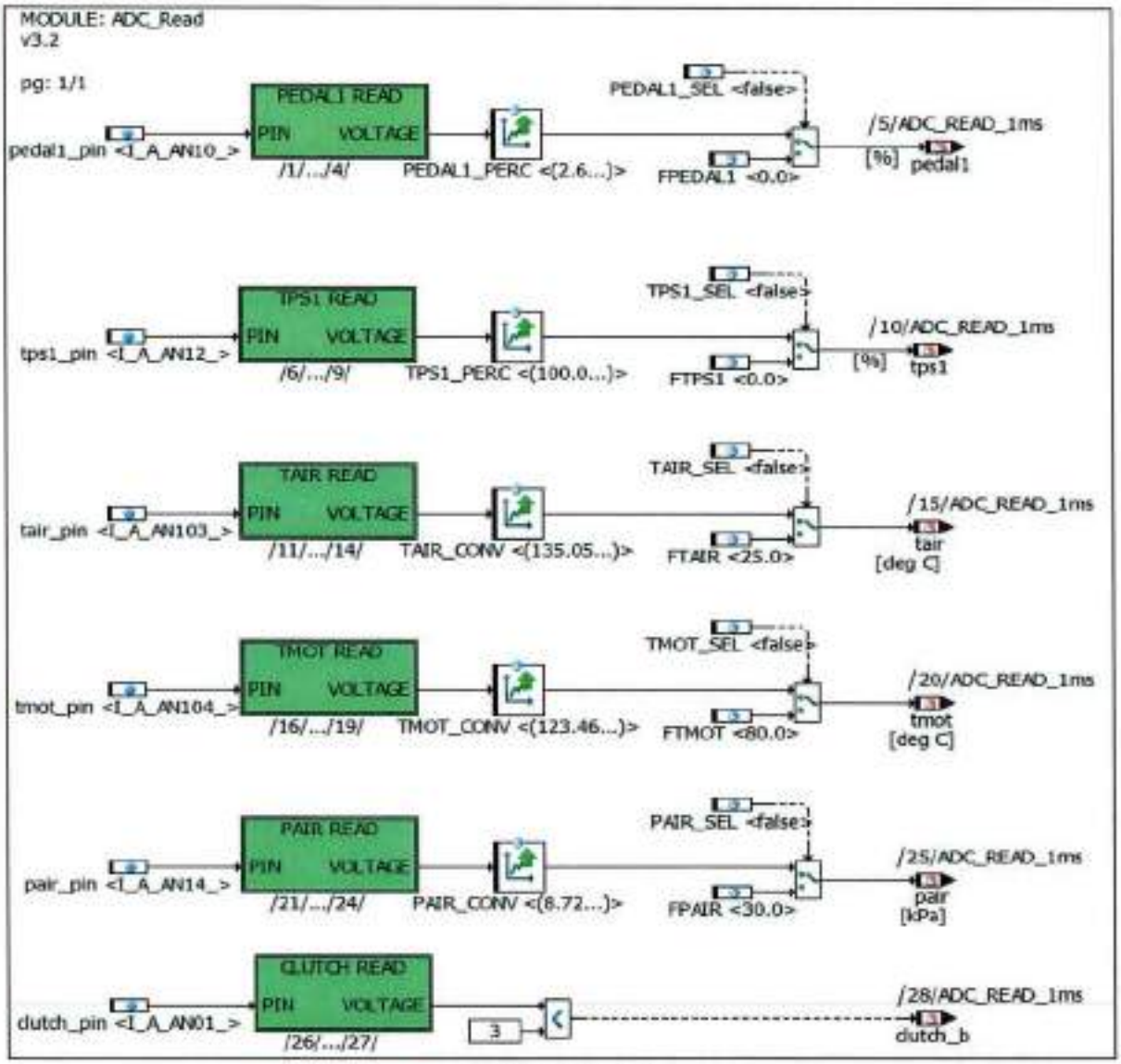
Fonte: o Autor

Figura 52 – Subsistema "DWELL TIME"



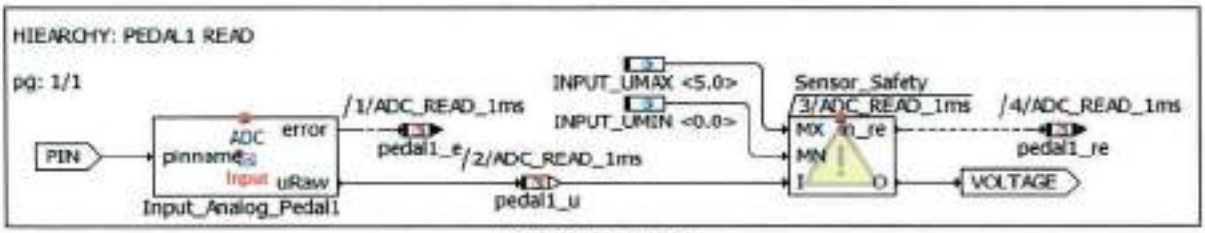
Fonte: o Autor

Figura 53 – Módulo "ADC_Read"



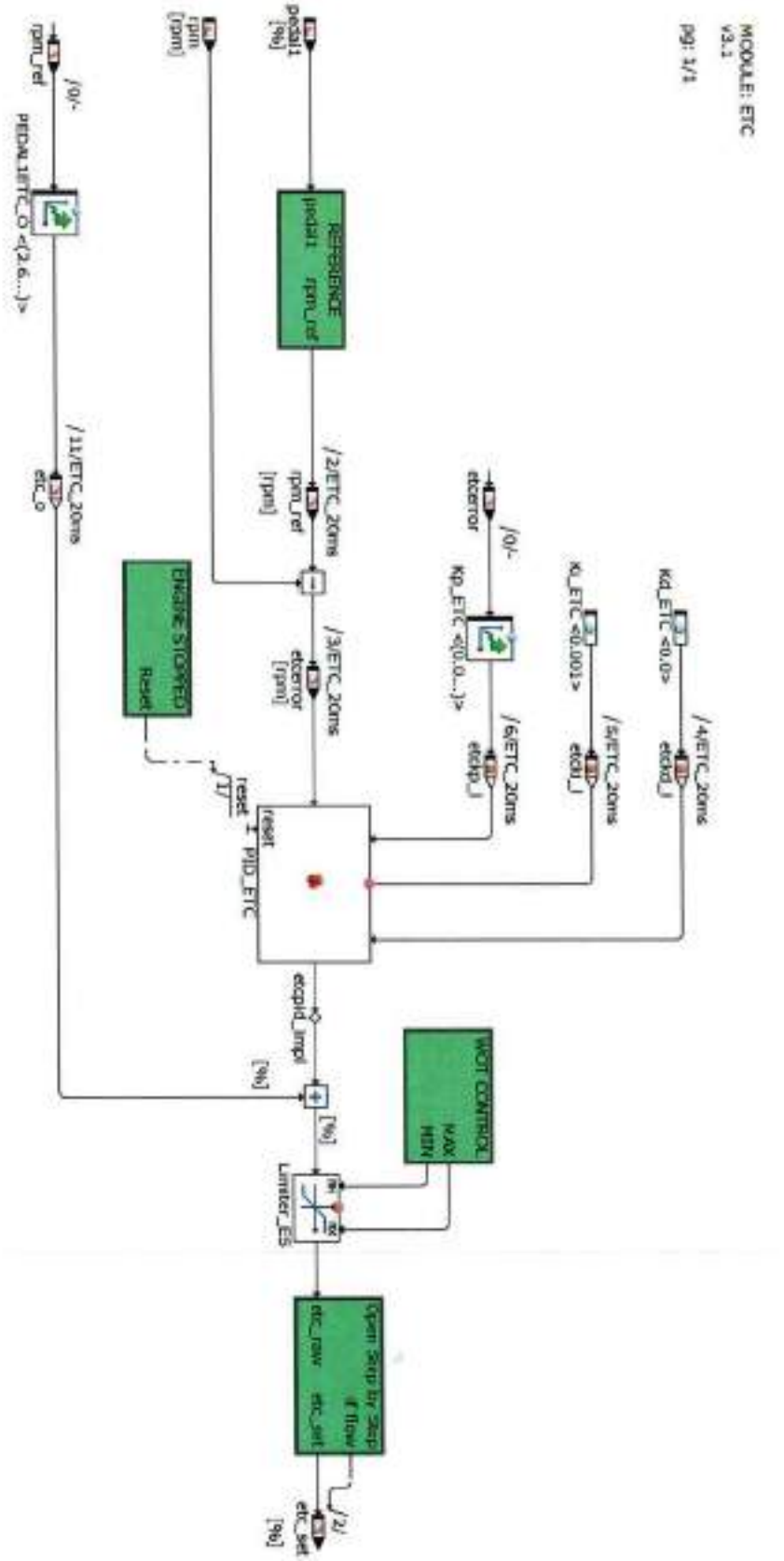
Fonte: o Autor

Figura 54 – Subsistemas "... READ"



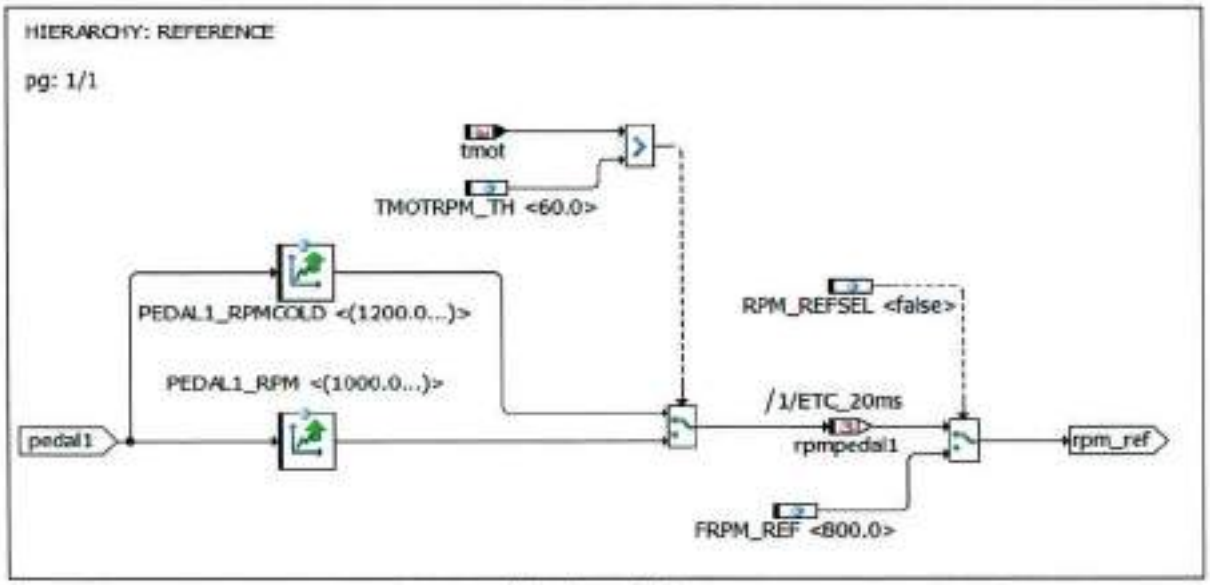
Fonte: o Autor

Figura 55 – Módulo "ETC"



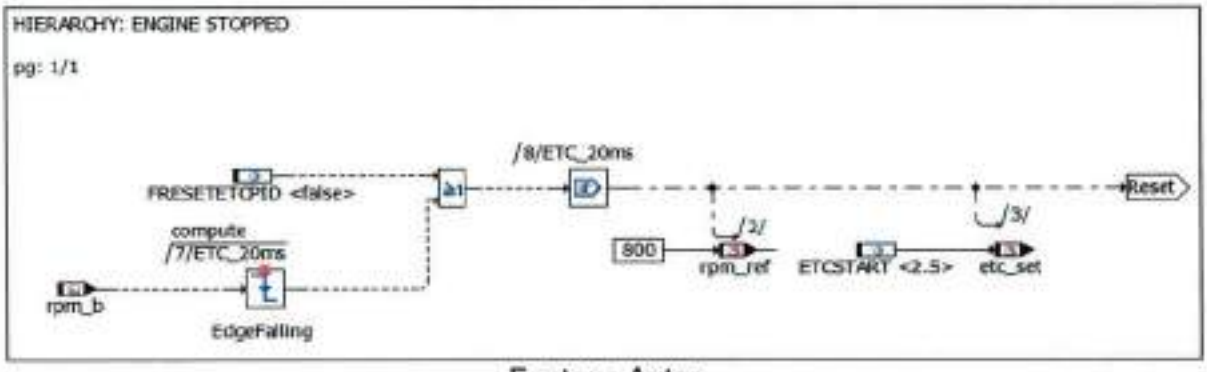
Fonte: o Autor

Figura 56 – Subsistema "REFERENCE"



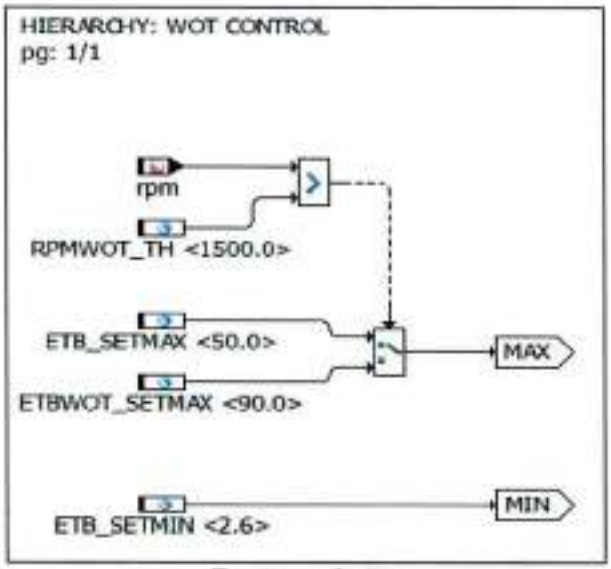
Fonte: o Autor

Figura 57 – Subsistema "ENGINE STOPPED"



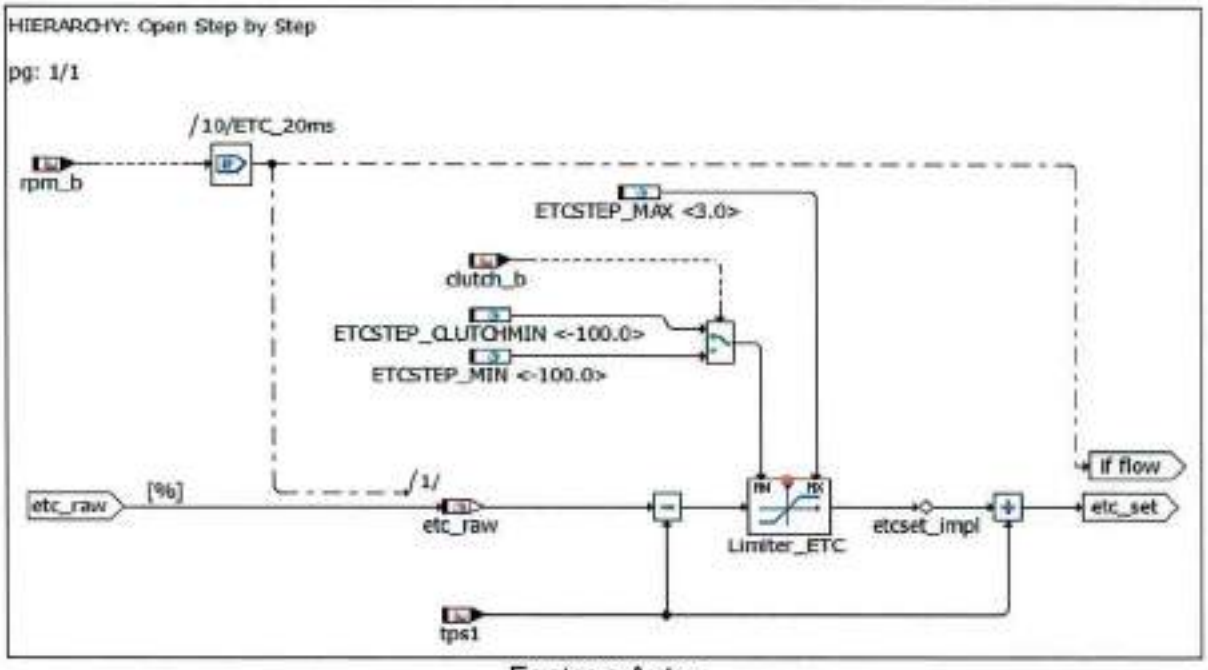
Fonte: o Autor

Figura 58 – Subsistema "WOT CONTROL"



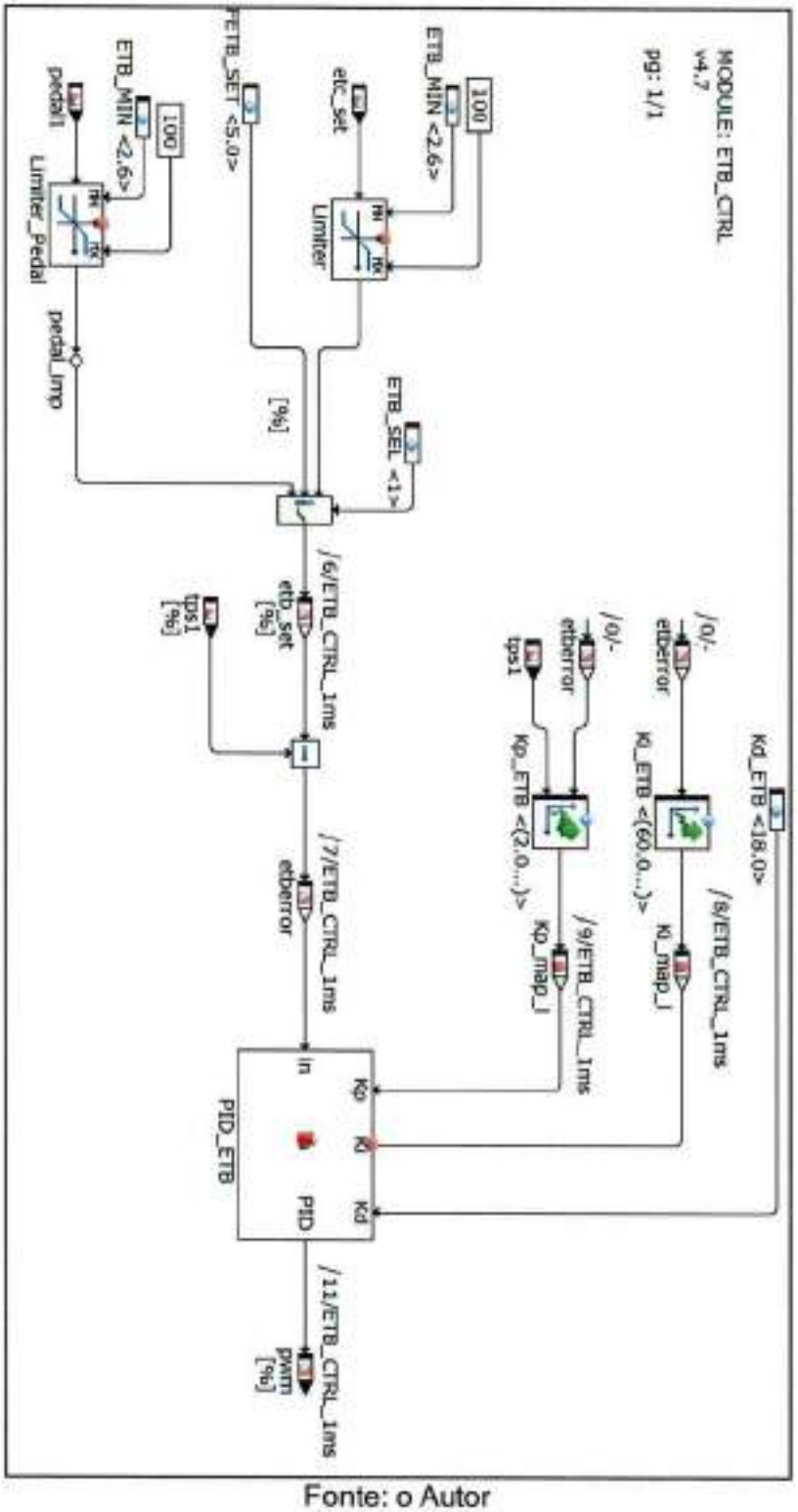
Fonte: o Autor

Figura 59 – Subsistema "Open Step by Step"



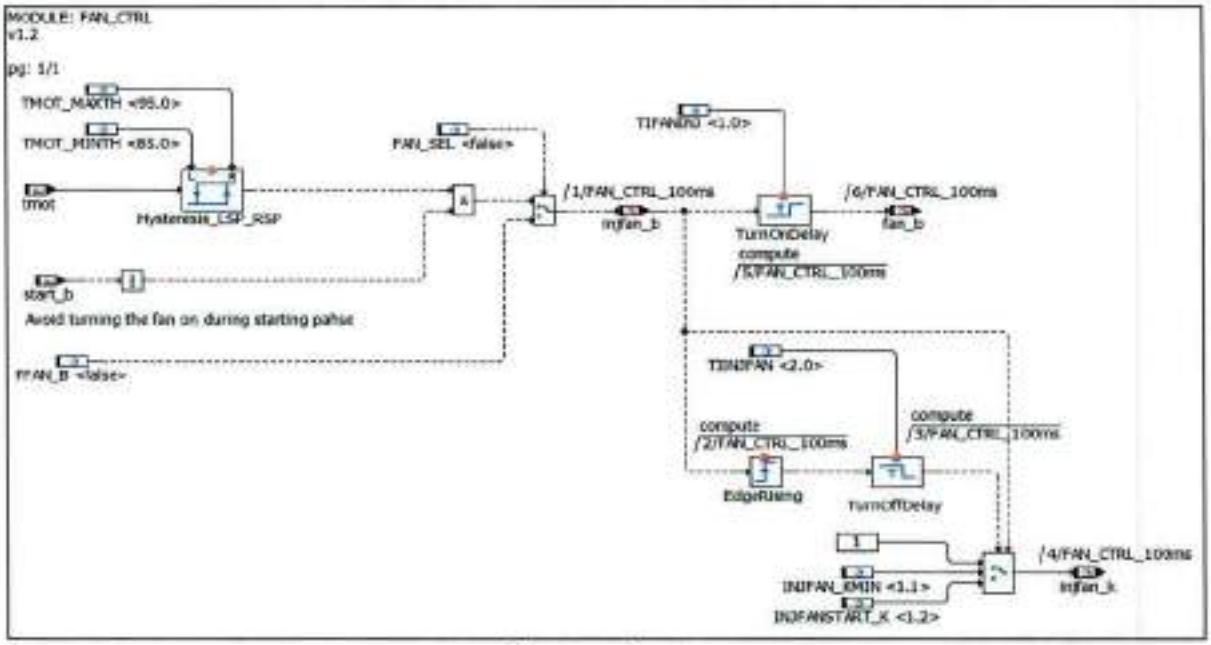
Fonte: o Autor

Figura 60 – Módulo "ETB_CTRL"



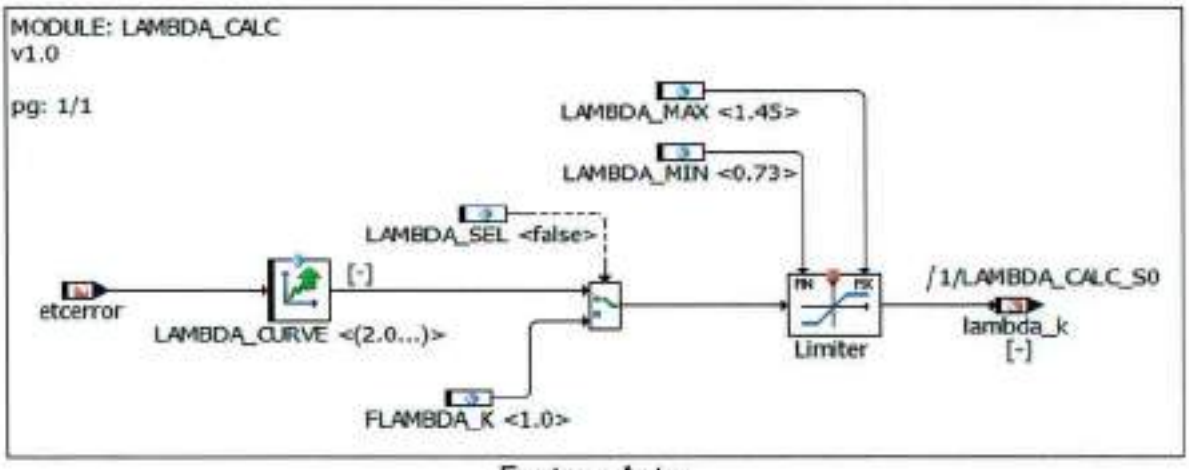
Fonte: o Autor

Figura 61 – Módulo "FAN_CTRL"



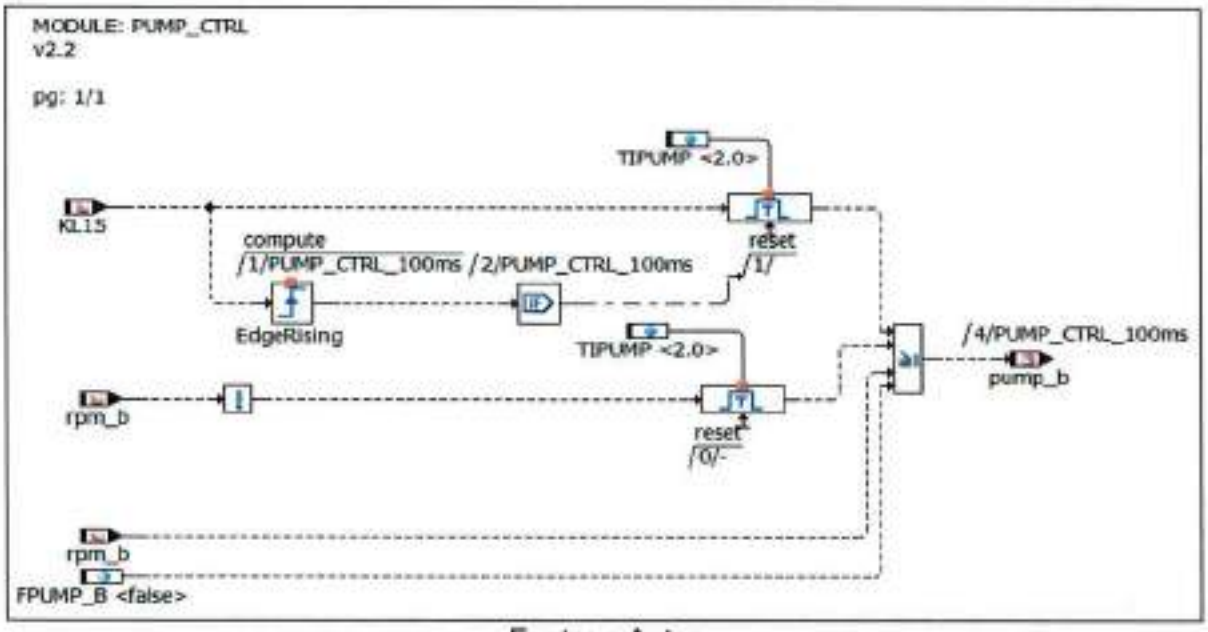
Fonte: o Autor

Figura 62 – Módulo "LAMBDA_CALC"



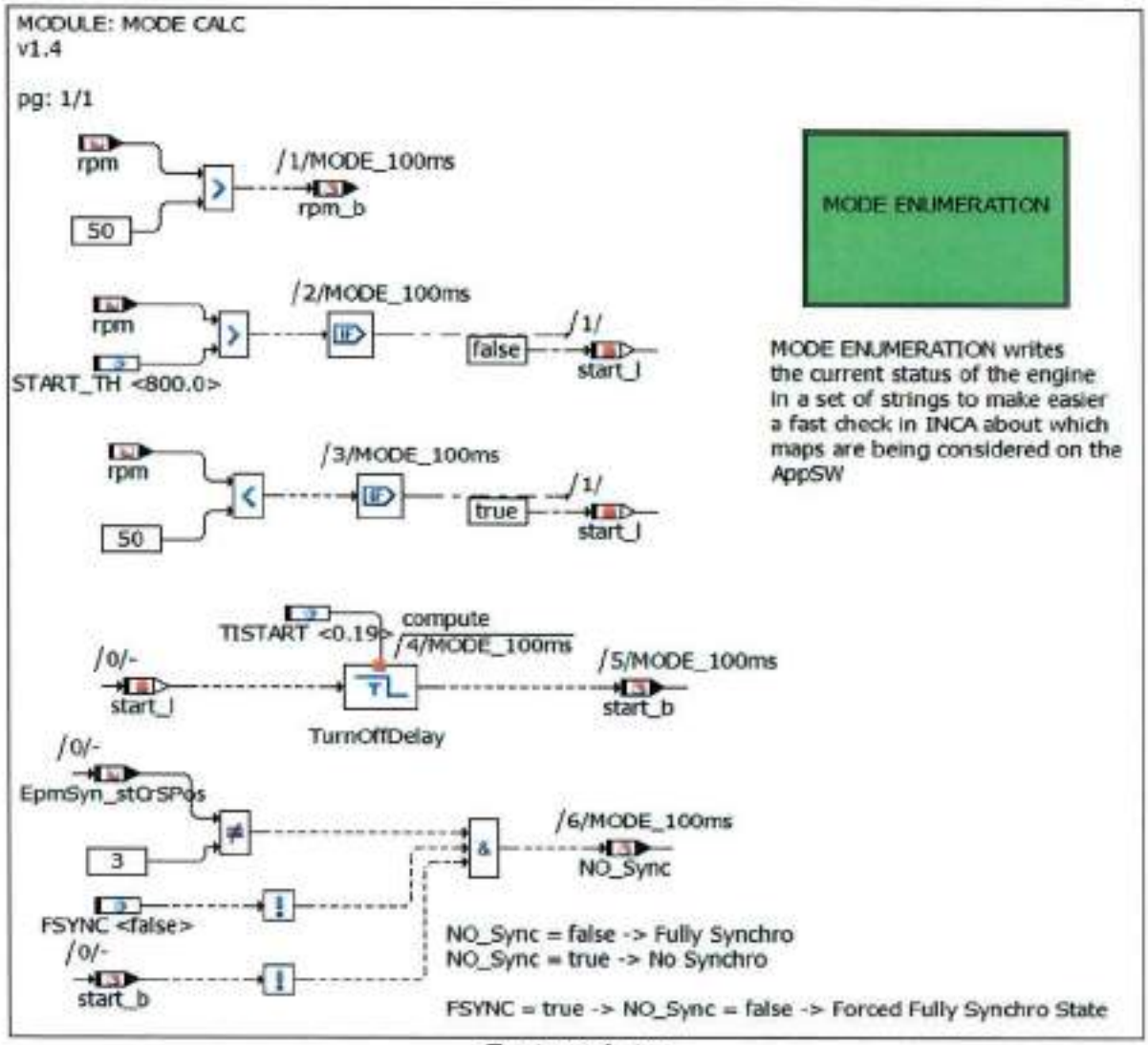
Fonte: o Autor

Figura 63 – Módulo "PUMP_CTRL"



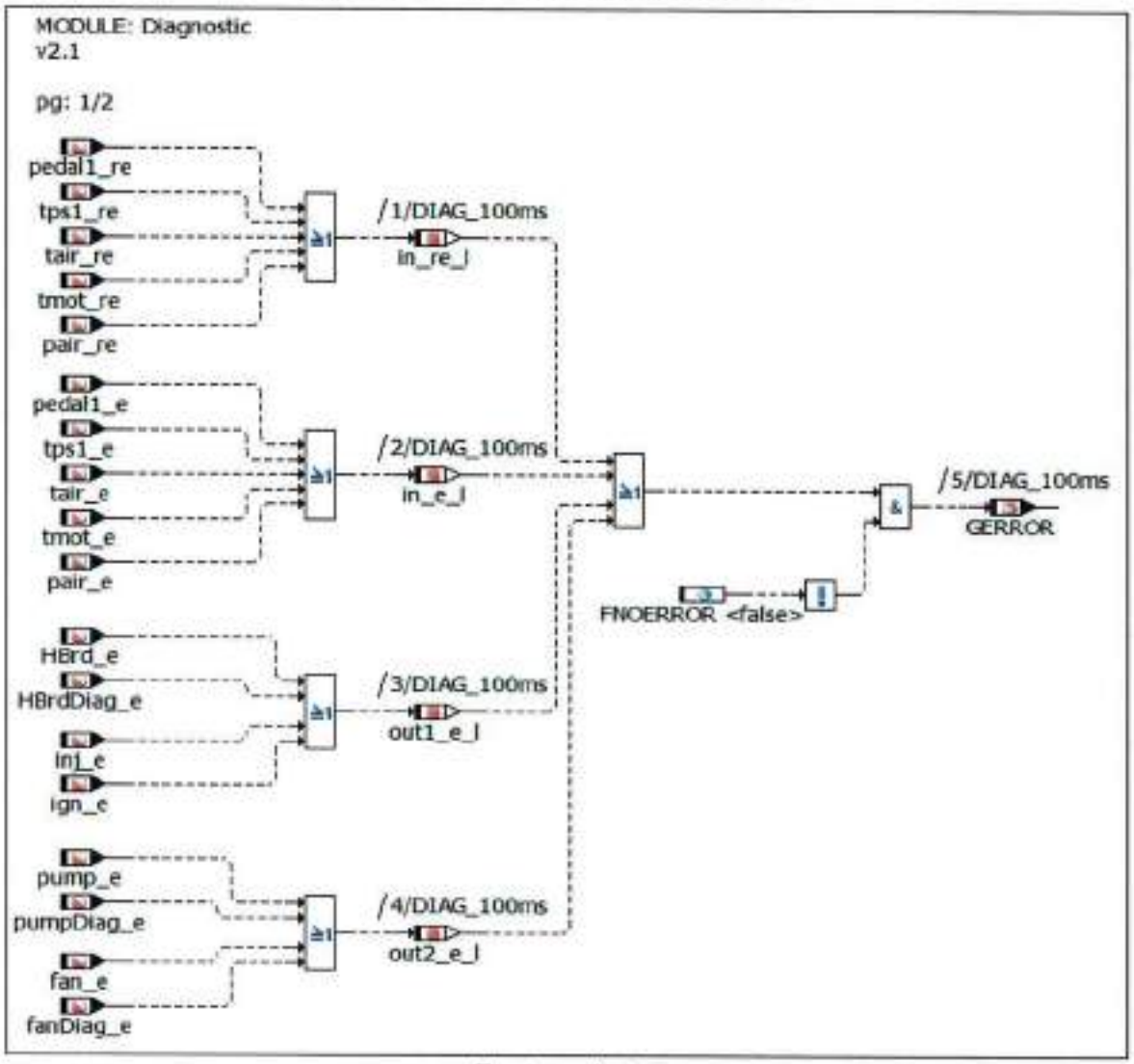
Fonte: o Autor

Figura 64 – Módulo "MODE_CALC"



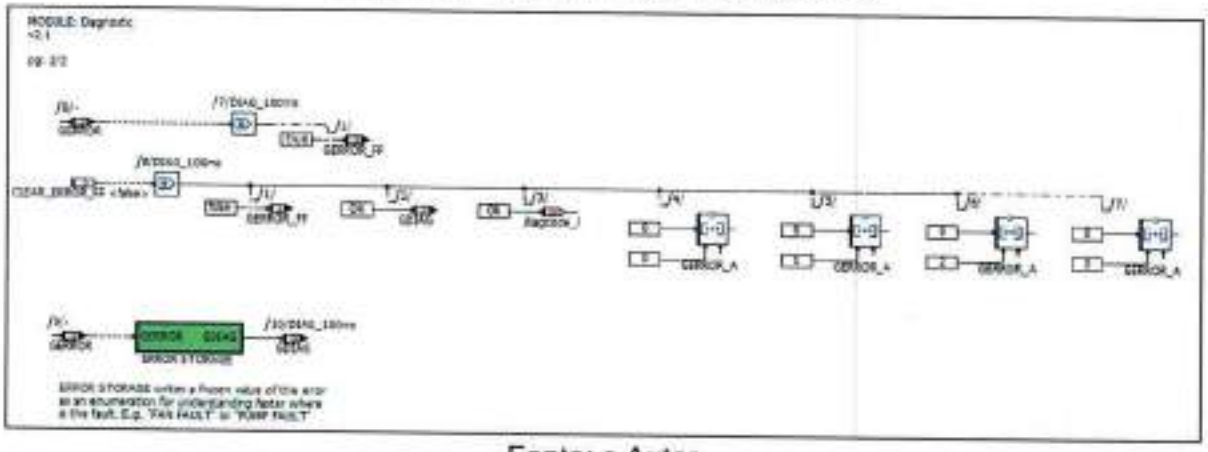
Fonte: o Autor

Figura 65 – Módulo "Diagnostic" parte 1



Fonte: o Autor

Figura 66 – Módulo "Diagnostic" parte 2

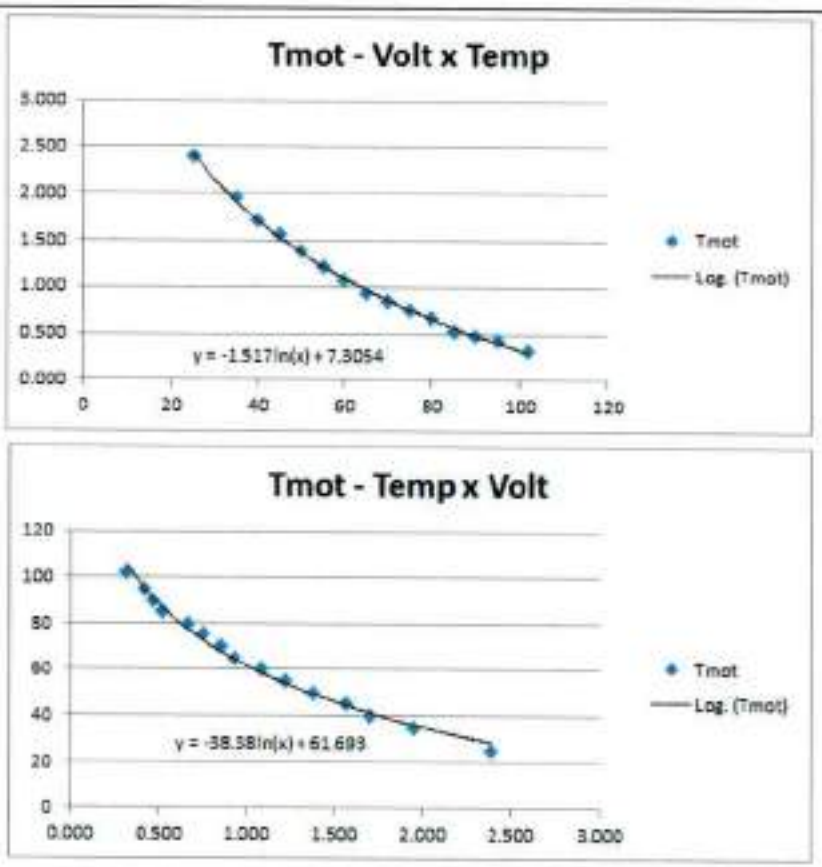


Fonte: o Autor

APÊNDICE B – Curva dos sensores

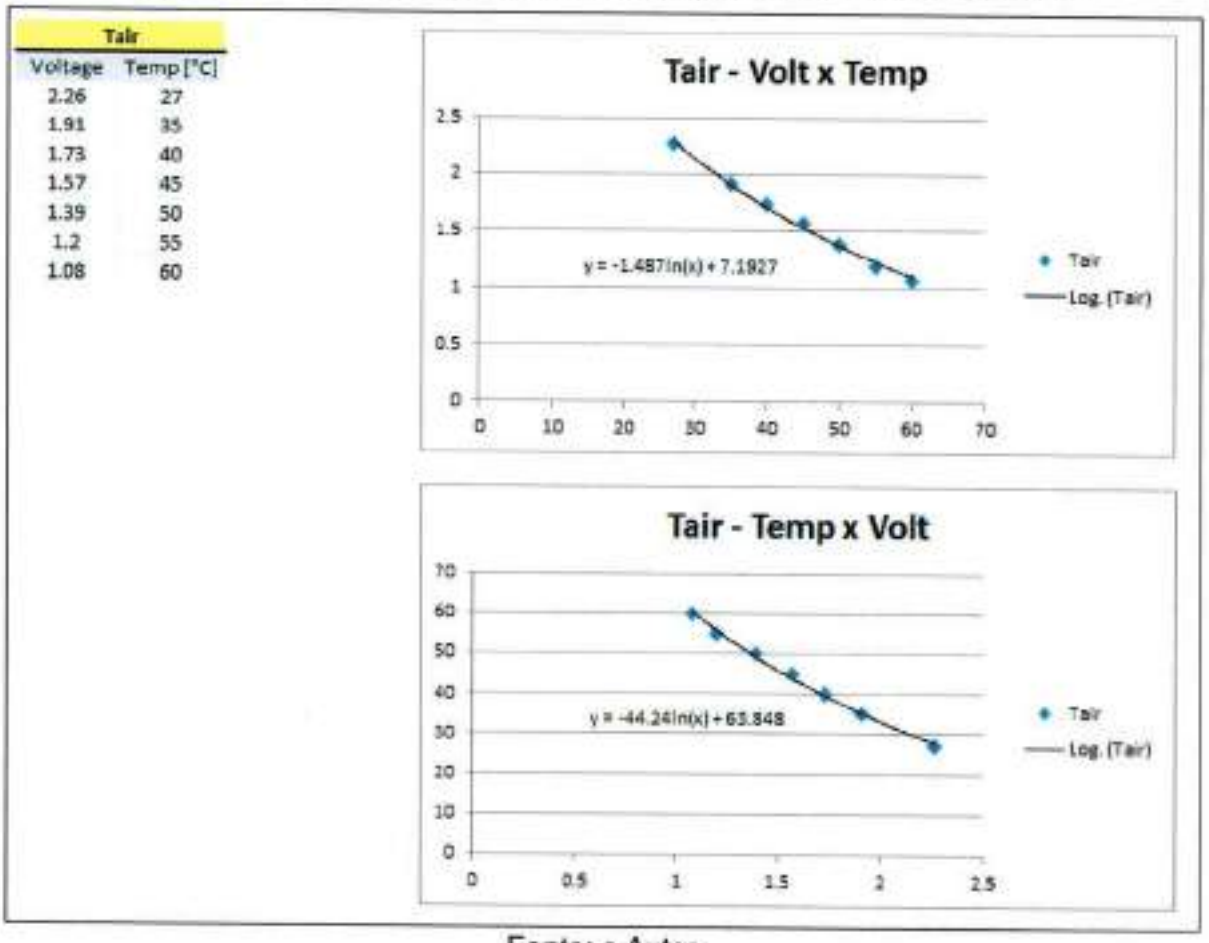
Figura 67 – Curva do sensor NTC de temperatura do motor

Tmot	
Voltage	Temp [°C]
0.320	102
0.420	95
0.470	90
0.520	85
0.670	80
0.750	75
0.850	70
0.930	65
1.080	60
1.220	55
1.380	50
1.570	45
1.710	40
1.950	35
2.390	25



Fonte: o Autor

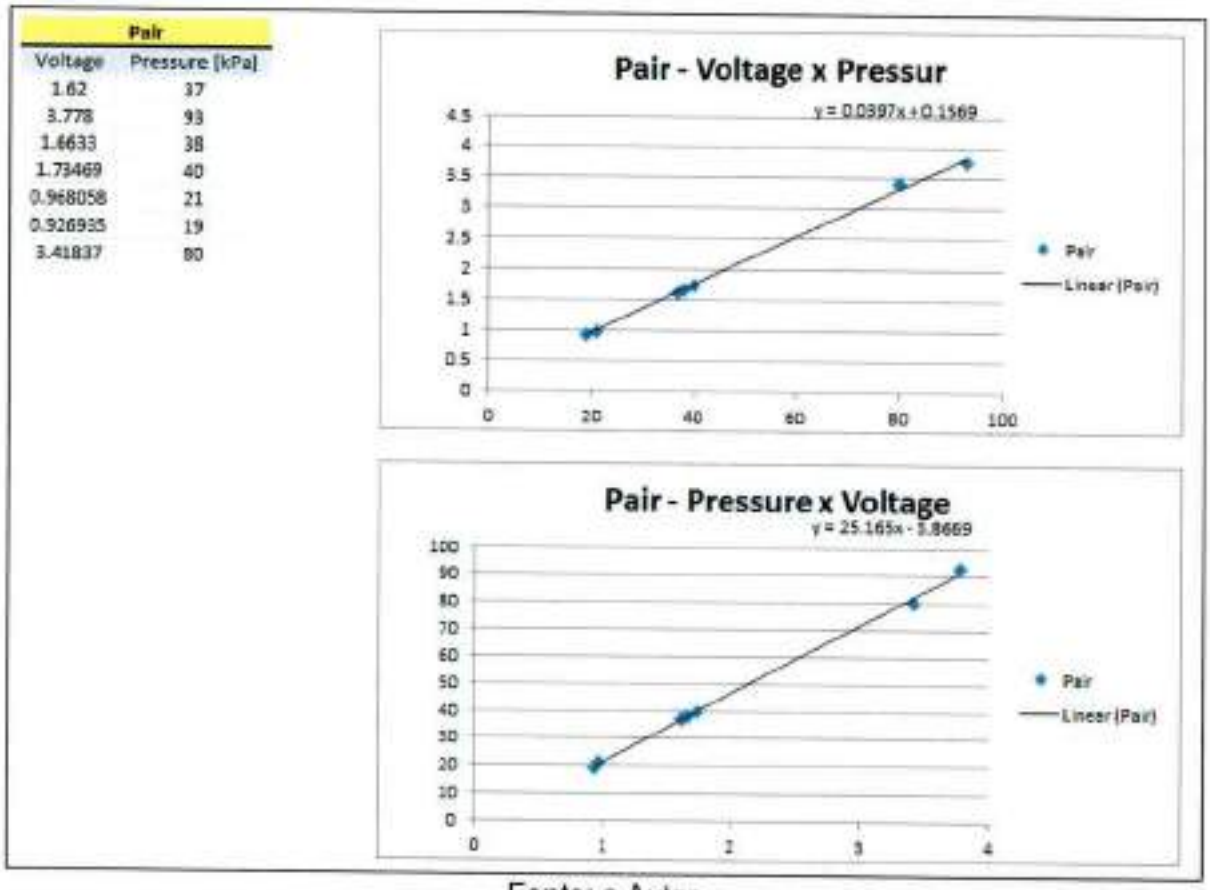
Figura 68 – Curva do sensor NTC de temperatura do ar de admissão



Tair	
Voltage	Temp [°C]
2.26	27
1.91	35
1.73	40
1.57	45
1.39	50
1.2	55
1.08	60

Fonte: o Autor

Figura 69 – Curva do sensor MAP



Pair	
Voltage	Pressure [kPa]
1.62	37
3.778	93
1.6633	38
1.73469	40
0.968058	21
0.926935	19
3.41837	80

Fonte: o Autor

Figura 70 – Curvas de conversão dos sensores

CALIBRATION MAPS																					
Tmot	5	10	20	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100	105	110	120
Voltage	4.86	3.81	2.76	2.15	1.91	1.71	1.53	1.37	1.23	1.09	0.97	0.86	0.76	0.66	0.57	0.48	0.40	0.32	0.25	0.17	0.04
Tair	5	10	20	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100	105	110	120
Voltage	4.80	3.77	2.74	2.14	1.91	1.71	1.53	1.38	1.23	1.10	0.99	0.88	0.77	0.68	0.59	0.50	0.42	0.34	0.27	0.20	0.07
Pair	10	20	30	40	50	60	70	80	90	100	110	120	130								
Voltage Raw	0.55	0.95	1.35	1.74	2.14	2.54	2.94	3.33	3.73	4.13	4.52	4.92	5.00								

Fonte: o Autor

APÊNDICE C – Pinout do Projeto Otto III

Figura 71 – Pinout usado em Otto III

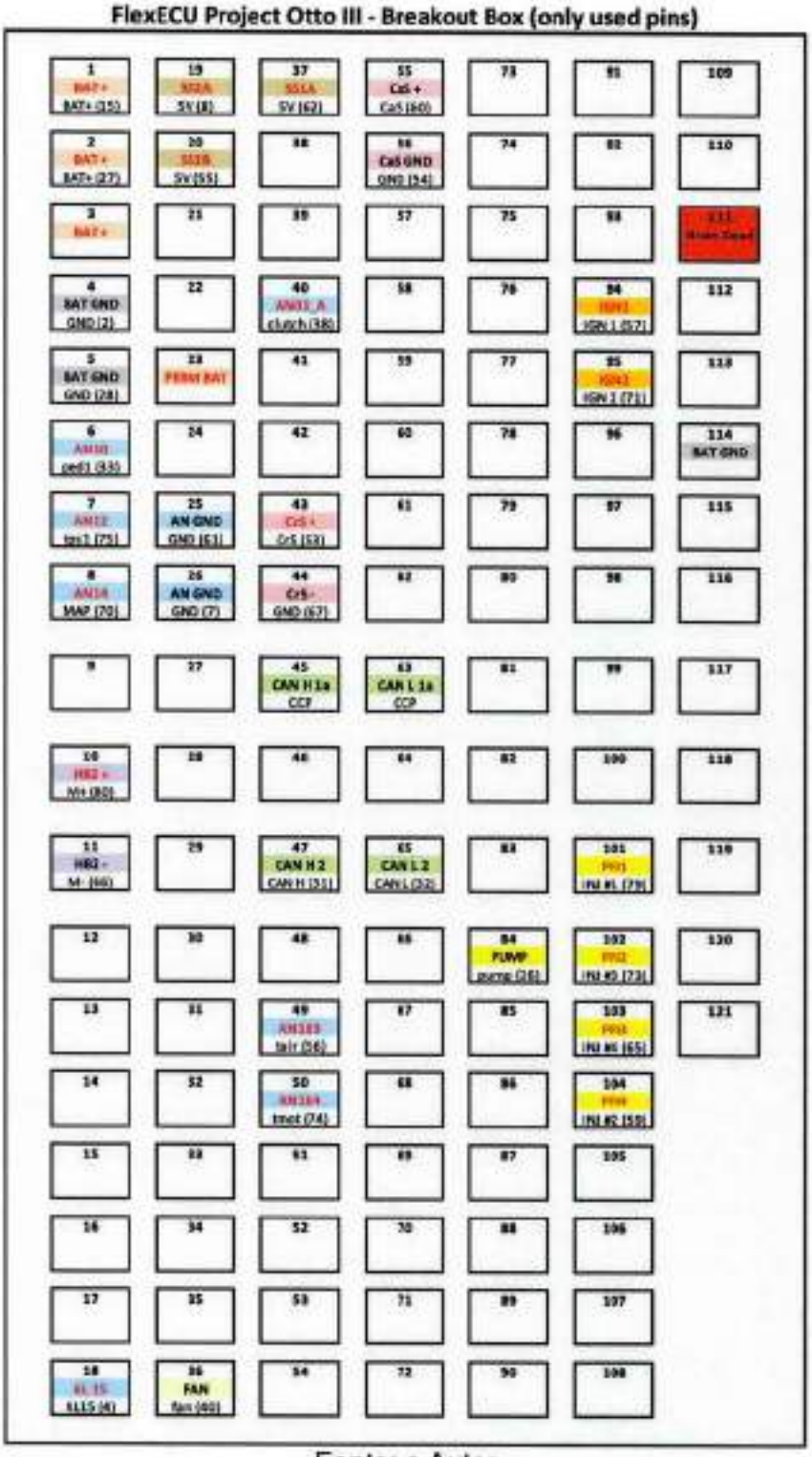
FlexECU - Project Otto III (only used pins)

Type	I/O	Pin Name	Pin No.	BoB Pin	Application
ECU Supply	Input	MR BAT +	K003	1	Supply Voltage from Main Relay
	Input	MR BAT +	K005	2	Supply Voltage from Main Relay
	Input	MR BAT +	K006	3	Supply Voltage from Main Relay
	Input	PERM BAT +	K086	23	Permanent BAT + (Term 30)
	Input	BAT -	K001	4	ECU Main Ground
	Input	BAT -	K002	5	ECU Main Ground
	Input	BAT -	K004	114	ECU Main Ground
Analog Sensors	Input	I_A_AN10	A084	6	Pedal Sensor 1
	Input	I_A_AN12	A063	7	ETB Position 1
	Input	I_A_AN14	K042	8	Manifold Absolute Pressure
	Input	I_A_AN103	K013	49	Water Temperature
	Input	I_A_AN01	A018	40	Clutch Singal
	Input	I_A_AN104	K030	50	Air Temperature
EPM	Input	I_F_DF08	A095	43	Crankshaft speed sensor signal
	Input	G_R_DF09	A096	44	Crankshaft speed sensor ground
	Input	I_F_DF01	A034	55	Camshaft speed sensor signal 1
	Input	G_R_DF	A036	56	Camshaft speed sensor ground
Digital Inputs	Input	I_S_DIG02	A035	111	Digital Input 2 and Brain Dead Pin*
	Input	I_S_T15	K050	18	Terminal 15 (switched BAT+)
Sensors GND	Input	G_R_AN02	A081	25	Analog/Digital Sensor GND
	Input	G_R_AN03	A060	26	Analog/Digital Sensor GND
H-Bridge Outputs	Output	O_T_DCPOS2	A043	10	H-Bridge Positive 2 for ETB
	Output	O_T_DCNEG2	A044	11	H-Bridge Negative 2 for ETB
Digital Outputs	Output	O_S_RL09	K040	84	Fan
	Output	O_T_RL14	K012	100	Pump
Ignition Power Stage	Output	O_P_IGN1	A051	94	Ignition Coil - Cylinder #1
	Output	O_P_IGN2	A072	95	Ignition Coil - Cylinder #2
	Output	G_R_IGN	A076	98	Ignition GND
5V Sensor Supply	Output	V_V_5VSS1A	A083	37	5V output for Sensor Supply
	Output	V_V_5VSS2A	A021	19	5V output for Sensor Supply
	Output	V_V_5VSS2B	A020	20	5V output for Sensor Supply
CAN Interface	-	B_D_CANH1a	K062	45	CAN High 1a - for CCP and Flash
	-	B_D_CANL1a	K063	63	CAN Low 1a - for CCP and Flash
	-	B_D_CANH2	K077	47	CAN High 2 - for Cluster
	-	B_D_CANL2	K078	65	CAN Low 2 - for Cluster
Injectors Output	Output	O_T_RL05	K072	101	PFI #1
	Output	O_T_RL06	K020	102	PFI #2
	Output	O_T_RL07	K036	103	PFI #3
	Output	O_T_RL08	K055	104	PFI #4

*Brain Dead Recovery Pin: **Connect this pin (Digital 02) to the GND and restart the ECU to enter ReFlash Recovery Mode**

Fonte: o Autor

Figura 72 – Breakout Box utilizada



Fonte: o Autor

Figura 73 – Conector da FlexECU e as conexões feitas no Otto III

Fonte: o Autor

APÊNDICE D – Nomenclatura das Variáveis

Tabela 17 – Nomenclatura das Variáveis

Prefix	Description	Variable	Related to	Sufix	Description
t	Temperature	rpm	Engine Speed	_b	Boolean
ti*	Time*	map	Main Calibration Maps	_e	Error bit
f	Forced Value	error	Controller Error	_d	Diagnosis
p	Pressure	pump	Fuel Pump	_k	Factor
r	Resistance	fan	Fan	_o	Offset
s	Safety	KL15	Key Line 15 (on)	_u	Voltage
		pedal	Accelerator Pedal	_safe	Safety Value
		tps	Throttle Position Sensor	_det	Detection
		pwm	H-Bridge PWM for ETB	_ctrl	Controller
		GERROR	Global Error Flag	_max	Maximum Value
		etb	Electronic Throttle Body	_min	Minimum Value
		Diag	Diagnosis Enum Block	_l	Local Variable
		hbrd	H-Bridge	_set	Setpoint
		clutch	Clutch Pedal	_sel	Selector
		etc	Eletronic Throttle Controller	_op	Opening
		air	Intake Air	_cl	Closing
		atm	Atmosphere	_re	Range Error
		batt	Battery	_ff	Freeze Frame
		mot	Engine	_pin	Pin number
		lambda	Lambda	_th	Threshold
		ang	Angle	_cut	Cutoff
		dwell	Dwell	_ref	Reference
		start	Starting Phase	_impl	Implentation
		inj	Injection	_conv	Conversion
		ign	Ignition	_pin	Pin number
		corr	Correction	_raw	Raw Value (not delayed)
		kp	Proportional Factor	_curve	Curve 1-D for Open Loop
		ki	Integral Factor		
		kd	Differential Factor		
		pid	Outcome from PID		
		wot	Wide Open Throttle		
		step	Step		

* The only exception is "tinj" which means Injection Time. So the two "i" were merged to avoid having double "i"

Fonte: o Autor