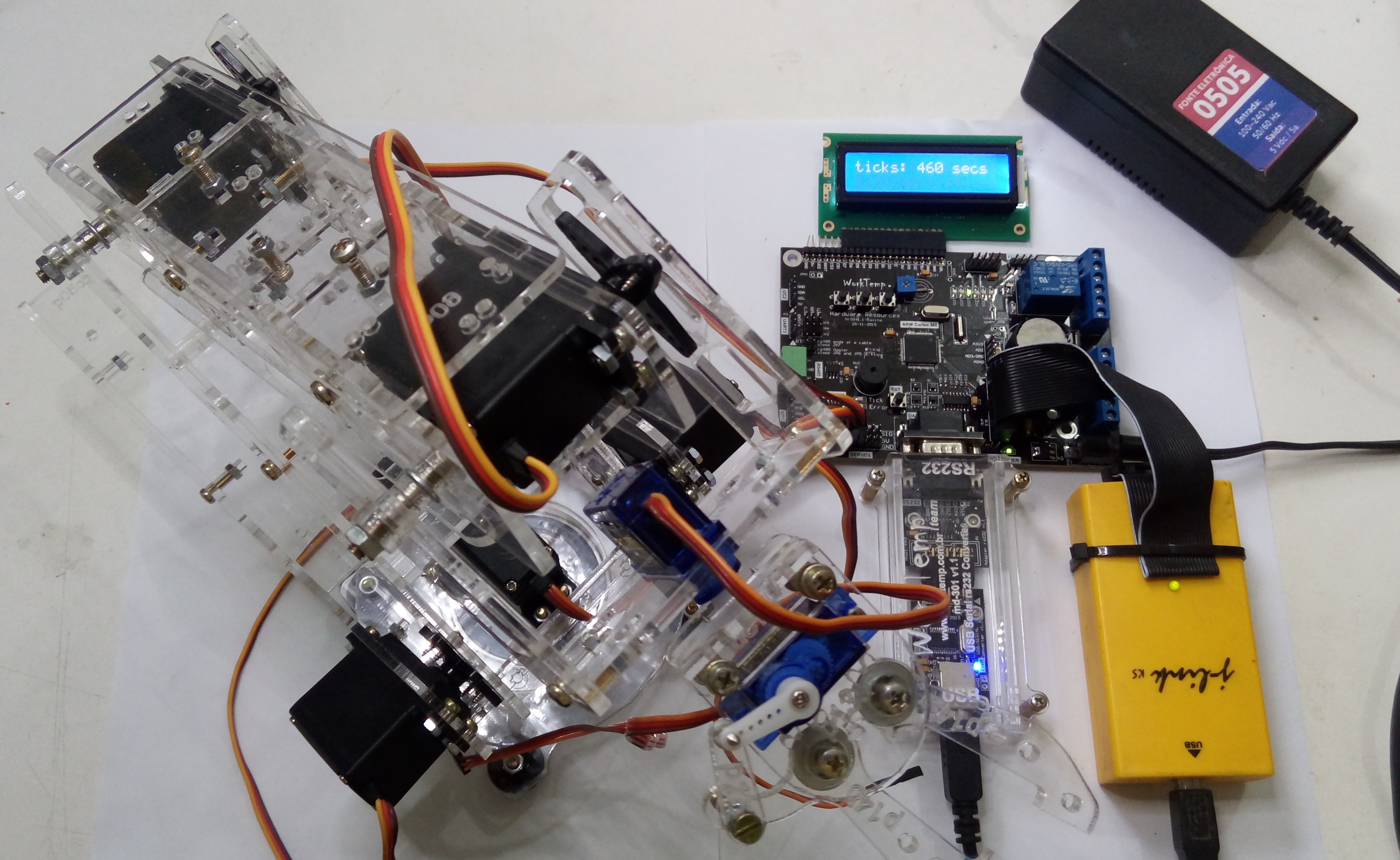
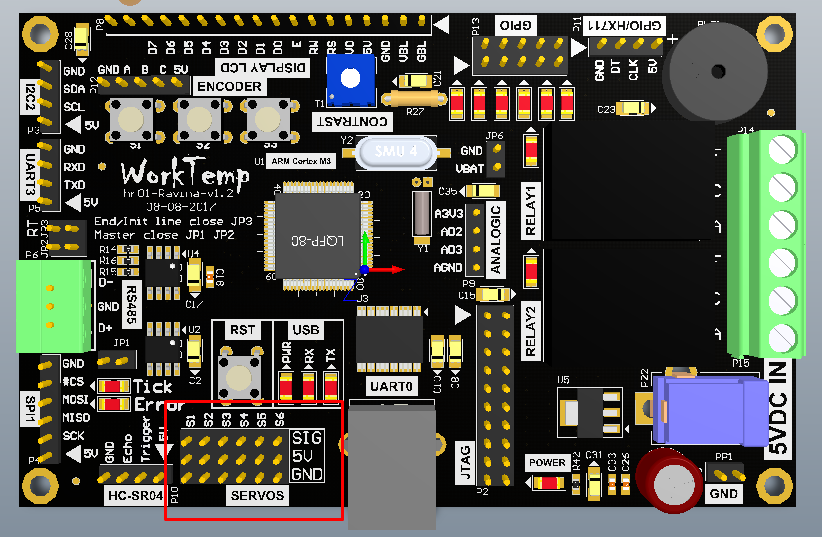
**Usando Servos Motores**

Neste exemplo vamos controlar até seis servos motores de aeromodelismo. Estes tipos de servos são muitos usados em mecatrônicas, e com isso é indispensável a sua implementação no kit ARM:



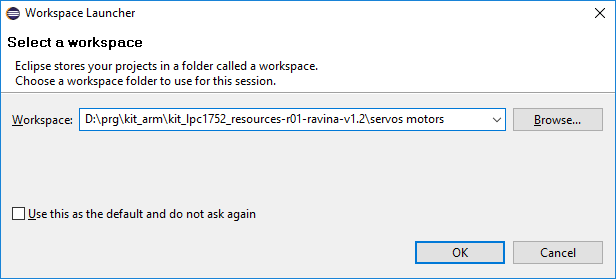
**Conexão com o Kit ARM**

No kit ARM há conetores específicos para conectar os servos, são os conectores S1 a S6. Nestes conectores contém o pino PWM de cada servo junto com pinos de alimentação. ***Importante***: A fonte de alimentação do kit ARM deve suportar o consumo de todos os servos ligados juntos:

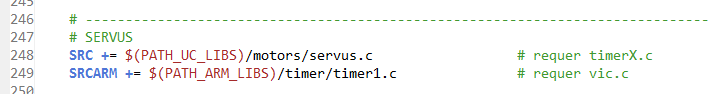


**Programa Exemplo**

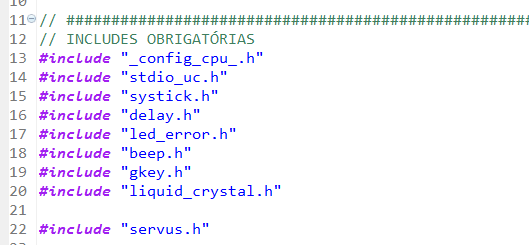
Abra o exemplo da pasta “servos motors” com o Eclipse:



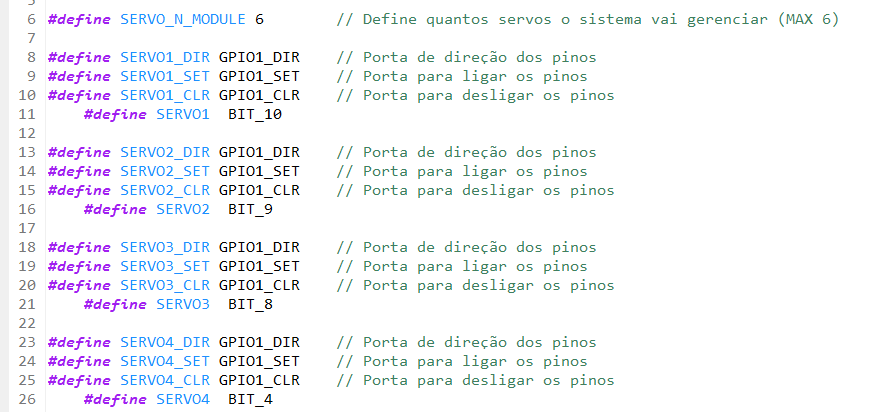
Foi criada a biblioteca servus.c que gerencia os sinais PWMs que são responsáveis para determinar os ângulos de todos os servos. A biblioteca servus.c requer um timer dedicado do processador para controlar os tempos dos PWMs e leituras de distancias dos ultrassons, logo foi usado o timer1 do ARM. No makefile, na secção de inserção de fontes, foi incluindo as fontes o servus.c e timer1.c:



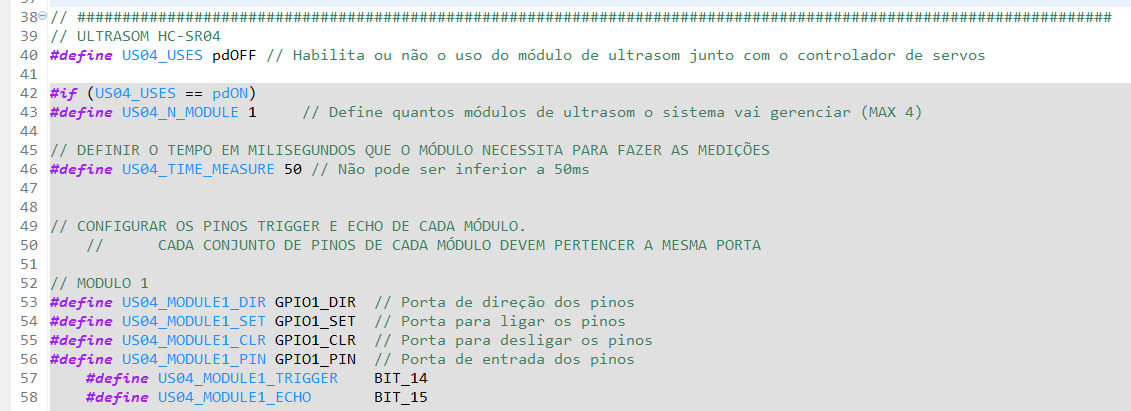
Em app.h foi adicionado a include do servus para usar as suas funções na aplicação:



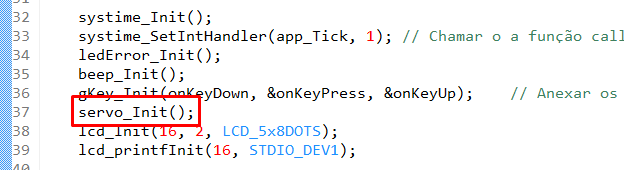
Em \_config\_cpu\_servus.h determinamos a quantidade de servos gerenciador vai controlar, e quais são os pinos IOs dos PWMs do ARM para os servos:



Ainda nesse arquivo de configuração podemos usar o recuso de controlar sensores de ultrassom modelo hc-sr04 e determinar os pinos ARM de controle. Isto porque geralmente usamos servos motores com outros tipos de sensores, e neste caso usamos a mesma estrutura do controlador de servos para controlar estes módulos de ultrassom:

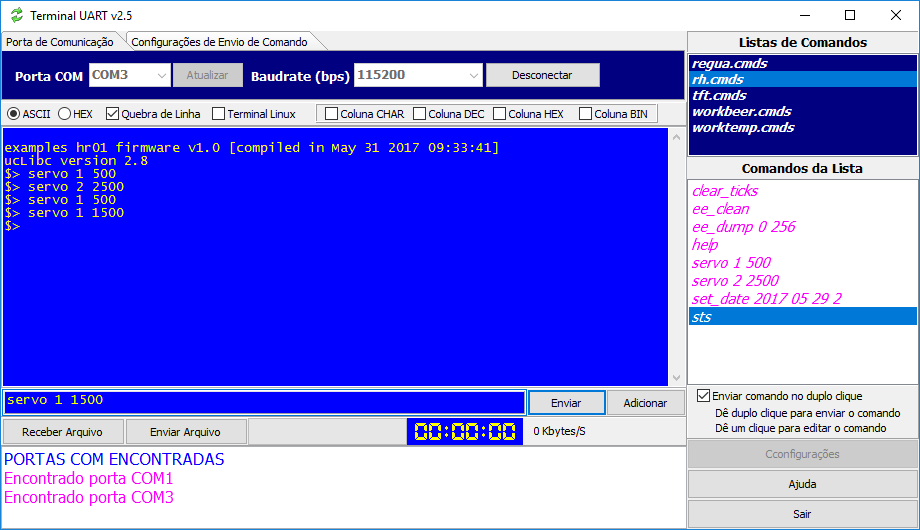


Em main.c dentro do int main(void), é inicializado o gerenciador servus:



**Compile e Execute**

Com tudo instalado e configurado, compile a aplicação, grave e execute no ARM. Após a execução todos os sevos receberão comandos para ficarem a 90° (1.5ms). Entre com o comando servo para determinar as novas posições, o primeiro parâmetro determina qual o servo que será ajustado, e o segundo parâmetro o tempo em milissegundos que servo vai assumir:



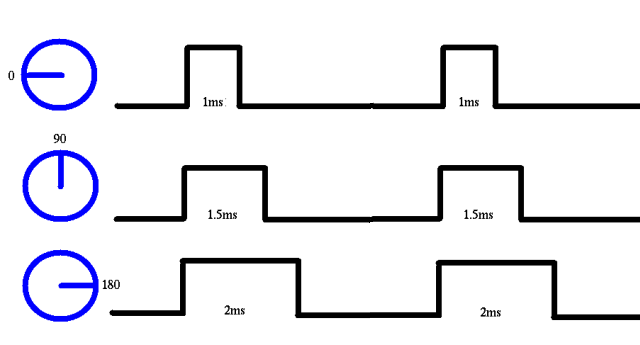
**Servos Motores**

Há muitos tipos de servo motores com diversos tamanhos, velocidades e torques, contudo, os servos motores que estamos usando são da área de hobby. Por ser um padrão no mundo dos hobbys há muitos acessórios para ajudar na construção de mecatrônica, como chassis, alavancas, eixos, rodas, link balls e muitos outros.



<https://engleandroalves.files.wordpress.com/2012/07/servo-4.jpg>

Esses servos já contém uma caixa de redução para oferecer um grande torque através de um pequeno motor DC. Dentro do servo há um controlador que determina em que ângulo o eixo deve ficar. O ângulo é fornecido externamente determinando o tempo que o sinal de controle deve ficar em alta, como mostrado na figura abaixo:



<http://www.lirtex.com/images/electronics/ServoMotors/ServoMotorControl.png>

Se o servo receber um pulso de 1ms, o controlador gira o seu eixo a 0°. Se o servo receber um pulso de 2ms, o controlador gira o seu eixo a 180°. Contudo, nem todos os servos se comportam dessa maneira, há servos que usam larguras de pulsos entre 0.5ms a 2500ms, ou outras faixas. Há também servos que são classificados de rotação continua.

O servo de rotação continua, utilizam os pulsos citados acima para ficar girando o seu eixo seja no sentido horário como anti-horário com graus de velocidades diferentes. Se o servo receber um pulso de 1.5ms, o controlador mantém o eixo parado, se o servo receber valores abaixo de 1.5ms, o controlador gira no sentido anti-horário, quando mais curto for o pulso mais veloz será a rotação. Se o servo receber valores acima de 1.5ms, o controlador gira no sentido horário, quando mais longo for o pulso mais veloz será a rotação.