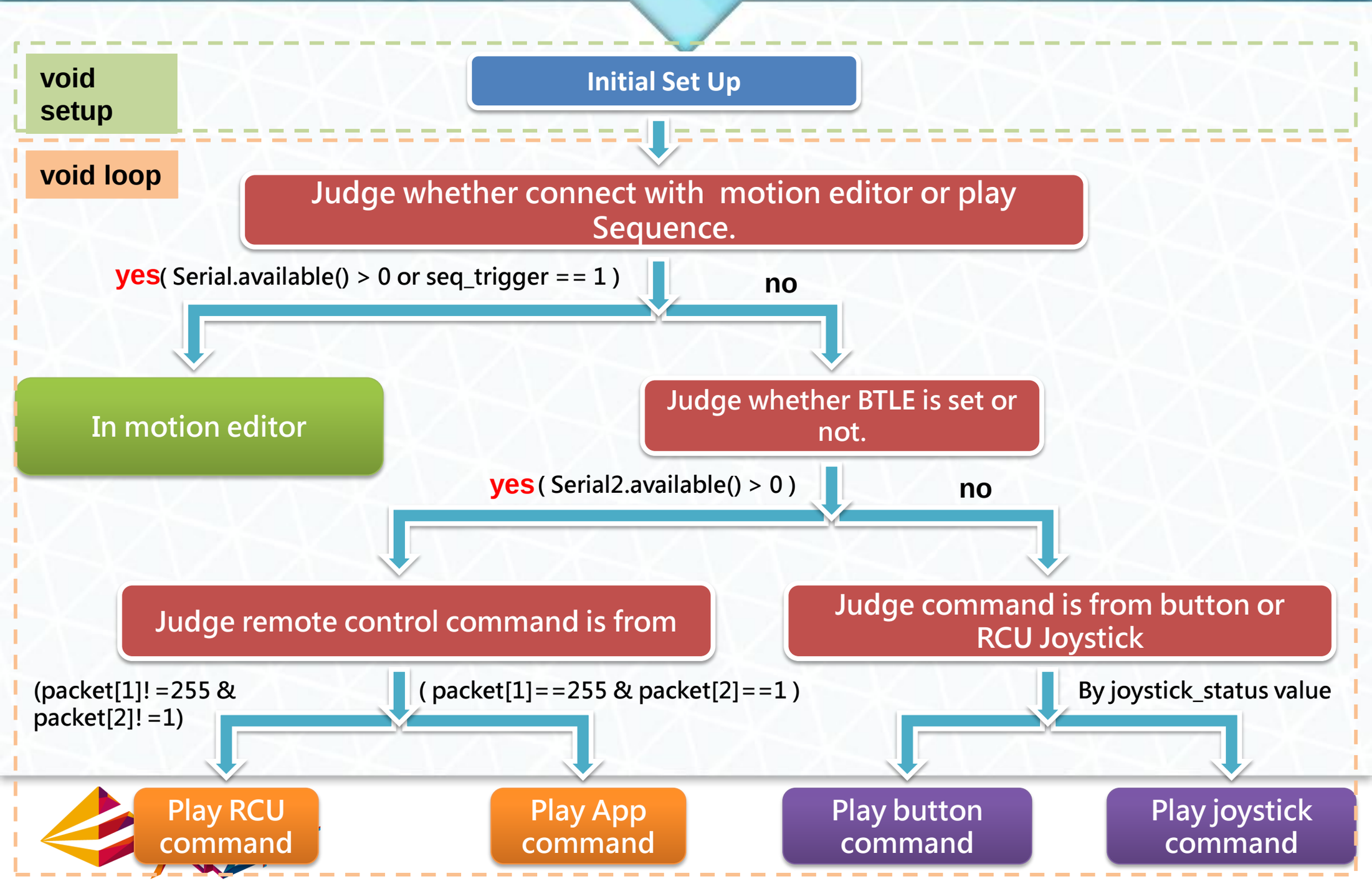
BOLIDE Y-01
Motion Editor programming instruction

Program structure



void setup()

```
//================== Set up ==================
void setup(){
  //Configure all basic setting
  Serial.begin(115200);
  AIM_Task_Setup();
  BT_Task_Setup();
  Speaker_Task_Setup();
  Eye_LED_Setup();
  Buzzer_Setup();
  Button_Setup();
  Analog_Input_Setup();
  Timer_Task_Setup();
  _enable_timer4();

  //Start motion
  LED_Task(2);
  Start_Music();
  G_SENSOR_Task_Setup();
  Initial_Pose_Setup();
  delay(1000);
  LED_Task(0);
}
```

Command	Function
AIM_Task_Setup()	Motor transmission setting
BT_Task_Setup()	Bluetooth transmission setting
Speaker_Task_Setup()	Audio transmission setting
Eye_LED_Setup()	Eyes LED pin setting
Buzzer_Setup()	Buzzer initial setting
Button_Setup()	Body buttons pin Initial setting
Analog_Input_Setup()	Voltage and IR sensor pin setting
Timer_Task_Setup()	Interrupt Timer setting
_enable_timer4()	Enable interrupt timer4
LED_Task()	LED mode
Start_Music()	Enable music
G_SENSOR_Task_Setup()	G sensor initial setting
Initial_Pose_Setup()	Robot motion to initial operation

void loop()

Editor Driver

```
void loop(){
  //USB Communcation motion
  if(Serial.available() > 0){
    Motion_Editor_Packet_Task();
  }

  //play sequence edited by motion editor
  else if(seq_trigger){
    Motion_Editor_Seq_Play();
  }
```

Command	Function
Motion_Editor_Packet_Task()	XYZrobot Editor driver
Motion_Editor_Seq_Play()	Play XYZrobot Editor Sequence

※Do not modify function above, in case XYZrobot Edito rerror.

BT connection

```
  else{
   //BT Communcation motion
   if(Serial2.available() > 0){
    BT_Packet_Task();
    if(BT_update){
      joystick_status[0] = packet[1];
      joystick_status[1] = packet[2];
      joystick_status[2] = packet[3];
      joystick_status[3] = packet[4];
```

Command	Function
BT_Packet_Task()	Read BT packet
joystick_status[]	RCU joystick status

void loop()

RCU

```
//==== RCU Command ====
if(packet[1]!=255 & packet[2]!=1){
  LED_Task(1);

  //Release Button
  if(packet[5] & RCU_mask_release){
    A1_16_TorqueOff(A1_16_Broadcast_ID);
    cb_BT();
  }

  //Bluetooth Button
  else if(packet[5] & RCU_mask_BT){
    cb_BT();
  }

  //Power Button
  else if(packet[5] & RCU_mask_power){
    XYZrobot.playSeq(DefaultInitial);
    while(XYZrobot.playing) XYZrobot.play();
    cb_BT();
  }
```

Judge condition

When pressed Release button

Torque off

When pressed BT pair button

When pressed power button

Play initial Sequence

void loop()

RCU

When pressed L1

```
//L1 Button
else if(packet[6] & RCU_mask_L1){
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_L1);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_L1);
}
```

When pressed L2

```
//L2 Button
else if(packet[6] & RCU_mask_L2){
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_L2);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_L2);
}
```

When pressed L3

```
//L3 Button
else if(packet[6] & RCU_mask_L3){
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_L3);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_L3);
}
```

When pressed R1

```
//R1 Button
else if(packet[5] & RCU_mask_R1){
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_R1);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_R1);
}
```

When pressed R2

```
//R2 Button
else if(packet[5] & RCU_mask_R2){
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_R2);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_R2);
}
```

When pressed R3

```
//R3 Button
else if(packet[5] & RCU_mask_R3){
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_R3);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_R3);
}
```

Command	Function
Adjustment_index	true，play get up Sequence；false，not thing happened
Falling_Task()	Check robot is standing or not
Action()	Play Sequence
Getup_Task()	Play get up Sequence

void loop()

RCU

When RCU on the left joystick to the right.

```
//LeftJoystick_Rightside
else if((packet[1]>155 & packet[2]>155 & packet[1]>packet[2])|(packet[1]>155 & packet[2]<95 & (packet[1]-155)>(95-packet[2]))|(packet[1]>155 & packet[2]>=95 & packet[2]<=155)){
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_LJR);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_LJR);
}
```

When RCU on the left joystick to the left.

```
//LeftJoystick_Leftside
else if((packet[1]<95 & packet[2]>155 & (95-packet[1])>(packet[2]-155))|(packet[1]<95 & packet[2]<155 & (95-packet[1])>(95-packet[2]))|(packet[1]<95 & packet[2]>=95 & packet[2]<=155)){
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_LJL);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_LJL);
}
```

void loop()

RCU

When RCU on the left joystick to the forward.

```
//LeftJoystick_Upside
else if((packet[1]>155 & packet[2]>155 & packet[1]<packet[2])|(packet[1]<95 & packet[2]>155 & (95-packet[1])<(packet[2]-155))|(packet[2]>155 & packet[1]>=95 & packet[1]<=155)){
  if(Adjustment_index){
    if (Falling_Task()==5){
      if (Avoidance_index & IR_SENSOR_Task() < 20){
       for(int i = 0; i<3; i++){
         tone(BUZZER_PIN,pgm_read_word_near(&obstacle_alarm_frq[i]));
         delay(250);
         noTone(BUZZER_PIN);
        }
      }
      else Action(RCU_LJU);
    }
     else Getup_Task(Falling_Task());
   }
   else Action(RCU_LJU);
}
```

Buzzer alert when obstacle is in front of IR sensor 20cm.

Command	Function
Avoidance_index	true，Buzzer alert ；false，not thing happened
IR_SENSOR_Task()	Detect obstacle

void loop()

RCU

When RCU on the left joystick to the back.

```
//LeftJoystick_Downside
else if((packet[1]>155 & packet[2]<95 & (packet[1]-155)<(95-packet[2]))|(packet[1]<95 & packet[2]<95 & (95-packet[1])<(155-packet[2]))|(packet[2]<95 & packet[1]>=95 & packet[1]<=155)){
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_LJD);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_LJD);
}
```

When RCU on the right joystick to the right.

```
//RightJoystick_Rightside
else if((packet[3]>155 & packet[4]>155 & packet[3]>packet[4])|(packet[3]>155 & packet[4]<95 & (packet[3]-155)>(95-packet[4]))|(packet[3]>155 & packet[4]>=95 & packet[4]<=155)){
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_RJR);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_RJR);
}
```

void loop()

RCU

When RCU on the right joystick to the left.

```
//RightJoystick_Leftside
else if((packet[3]<95 & packet[4]>155 & (95-packet[3])>(packet[4]-155))|(packet[3]<95 & packet[4]<95 & (95-packet[3])>(95-packet[4]))|(packet[3]<95 & packet[4]>=95 & packet[4]<=155)){
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_RJL);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_RJL);
}
```

When RCU on the right joystick to the forward.

```
//RightJoystick_Upside
else if((packet[3]>155 & packet[4]>155 & packet[3]<packet[4])|(packet[3]<95 & packet[4]>155 & (95-packet[3])<(packet[4]-155))|(packet[4]>155 & packet[3]>=95 & packet[3]<=155)){
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_RJU);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_RJU);
}
```

When RCU on the right joystick to the back.

```
//RightJoystick_Downside
else if((packet[3]>155 & packet[4]<95 & (packet[3]-155)<(95-packet[4]))|(packet[3]<155 & packet[4]<95 & (95-packet[3])<(95-packet[4]))|(packet[4]<95 & packet[3]>=95 & packet[3]<=155)){
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_RJD);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_RJD);
}

LED_Task(0);
BT_update = false;
```

void loop()

APP

Packet[3]	Function
101	Play initial Sequence
102	All smart servos Torque off
251	Feedback G sensor value
252	Feedback IR sensor value
253	Feedback firmware version
1~54	Play relative XYZrobot Editor Action List sequence

```
//=== App Command ===
else if(packet[1]==255 & packet[2]==1){
  LED_Task(3);
  if(packet[3] == 101){
    XYZrobot.playSeq(DefaultInitial);
    while(XYZrobot.playing) XYZrobot.play();
  }
  else if(packet[3] == 102) A1_16_TorqueOff(254);
  else if(packet[3] == 251) BT_Gsensor_Data();
  else if(packet[3] == 252) BT_IR_Data();
  else if(packet[3] == 253) BT_FW();
  else{
    if(Adjustment_index){
      if(Falling_Task()==5){
        if(packet[3] == 1){//WalkForward
          if(Avoidance_index & IR_SENSOR_Task() < 20){
            for(int i = 0; i<3; i++){
              tone(BUZZER_PIN,pgm_read_word_near(&obstacle_alarm_frq[i]));
              delay(250);
              noTone(BUZZER_PIN);
            }
          }
          else Action(packet[3]);
        }
      else Action(packet[3]);
      }
    else Getup_Task(Falling_Task());
    }
    else Action(packet[3]);
  }
  LED_Task(0);
  BT_update = false;
}
```

Judge condition

Judge whether play get up Sequence or buzzer alert during robot moving forward.

void loop()

Button

Button been pressed or not

```
else{
  if(joystick_status[0]<=155 & joystick_status[0]>=95 & joystick_status[1]<=155 & joystick_status[1]>=95 & joystick_status[2]<=155 & joystick_status[2]>=95 & joystick_status[3]<=155 & joy
    // Button task
     BUTTON_Task();
   }
```

RCU joystick

When RCU on the left joystick to the right.

```
if((joystick_status[0]>155 & joystick_status[1]>155 & joystick_status[0]>joystick_status[1])|(joystick_status[0]>155 & joystick_status[1]<95 & (joystick_status[0]-155)>(95-joystick_status[1])
 //LeftJoystick_Rightside
   if(Adjustment_index){
     if (Falling_Task()==5) Action(RCU_LJR);
     else Getup_Task(Falling_Task());
   }
   else Action(RCU_LJR);
}
```

When RCU on the left joystick to the left.

```
else if((joystick_status[0]<95 & joystick_status[1]>155 & (95-joystick_status[0])>(joystick_status[1]-155))|(joystick_status[0]<95 & joystick_status[1]<155 & (95-joystick_status[0])>(95-joyst
 //LeftJoystick_Leftside
  if(Adjustment_index){
    if (Falling_Task()==5) Action(RCU_LJL);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_LJL);
}
```

void loop()

RCU joystick

When RCU on the left joystick to the forward.

```
else if((joystick_status[0]>155 & joystick_status[1]>155 & joystick_status[0]<joystick_status[1])|(joystick_status[0]<95 & joystick_status[1]>155 & (95-joystick_status[0])<(joystick_status[1
 //LeftJoystick_Upside
  if(Adjustment_index){
    if(Falling_Task()==5){
      if(Avoidance_index & IR_SENSOR_Task() < 20){
        for(int i = 0; i<3; i++){
          tone(BUZZER_PIN,pgm_read_word_near(&obstacle_alarm_frq[i]));
          delay(250);
          noTone(BUZZER_PIN);
        }
      }
      else Action(RCU_LJU);
    }
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_LJU);
}
```

When RCU on the left joystick to the back.

```
else if((joystick_status[0]>155 & joystick_status[1]<95 & (joystick_status[0]-155)<(95-joystick_status[1]))|(joystick_status[0]<95 & joystick_status[1]<95 & (95-joystick_status[0])<(155-joys
 //LeftJoystick_Downside
  if(Adjustment_index){
    if(Falling_Task()==5) Action(RCU_LJD);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_LJD);
}
```

When RCU on the right joystick to the right.

```
else if((joystick_status[2]>155 & joystick_status[3]>155 & joystick_status[2]>joystick_status[3])|(joystick_status[2]>155 & joystick_status[3]<95 & (joystick_status[2]-155)>(95-joystick_stat
 //RightJoystick_Rightside
  if(Adjustment_index){
    if(Falling_Task()==5) Action(RCU_RJR);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_RJR);
}
```

void loop()

RCU joystick

When RCU on the right joystick to the left.

```
else if((joystick_status[2]<95 & joystick_status[3]>155 & (95-joystick_status[2])>(joystick_status[3]-155))|(joystick_status[2]<95 & joystick_status[3]<95 & (95-joystick_status[2])>(95-joysti
 //RightJoystick_Leftside
  if(Adjustment_index){
    if(Falling_Task()==5) Action(RCU_RJL);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_RJL);
}
```

When RCU on the right joystick to the forward.

```
else if((joystick_status[2]>155 & joystick_status[3]>155 & joystick_status[2]<joystick_status[3])|(joystick_status[2]<95 & joystick_status[3]>155 & (95-joystick_status[2])<(joystick_status[3]
 //RightJoystick_Upside
  if(Adjustment_index){
    if(Falling_Task()==5) Action(RCU_RJU);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_RJU);
}
```

When RCU on the right joystick to the back.

```
else if((joystick_status[2]>155 & joystick_status[3]<95 & (joystick_status[2]-155)<(95-joystick_status[3]))|(joystick_status[2]<155 & joystick_status[3]<95 & (95-joystick_status[2])<(
 //RightJoystick_Downside
  if(Adjustment_index){
    if(Falling_Task()==5) Action(RCU_RJD);
    else Getup_Task(Falling_Task());
  }
  else Action(RCU_RJD);
 }
}
```

Function

AIM_Task_Setup( ) ⇒ Setup Smart servo A1-16 transmissions baud rate and number

```
void AIM_Task_Setup(void){
  XYZrobot.setup(115200, 18);
}
```

BT_Task_Setup( ) ⇒ Setup BT Baud rate 9600

```
void BT_Task_Setup(void){
  Serial2.begin(9600);
}
```

Speaker_Task_Setup( ) ⇒ Setup **Audio Baud rate 115200；Setup pin of LED which on Audio PCB。**

```
void Speaker_Task_Setup(void){
  Serial3.begin(115200);
  pinMode(LSA_LED_BLUE_PIN, OUTPUT);
  pinMode(LSA_LED_GREEN_PIN, OUTPUT);
  pinMode(LSA_LED_RED_PIN, OUTPUT);
}
```

Eye_LED_Setup( ) ⇒ Setup pin of eyes **LED**

```
void Eye_LED_Setup(void){
  pinMode(LED_BLUE_PIN, OUTPUT);
  pinMode(LED_GREEN_PIN, OUTPUT);
}
```

Buzzer_Setup( ) ⇒ Setup buzzer pin

```
void Buzzer_Setup(void){
  pinMode(BUZZER_PIN, OUTPUT);
}
```

Function

Button_Setup( ) ⇨ Setup button pin

```
void Button_Setup(void){
  pinMode(BUTTON1_PIN, INPUT);
  pinMode(BUTTON2_PIN, INPUT);
  pinMode(BUTTON3_PIN, INPUT);
  pinMode(BUTTON4_PIN, INPUT);
}
```

Analog_Input_Setup( ) ⇨ Setup pin for voltage detect and pin of IR sensor

```
void Analog_Input_Setup(void){
  pinMode(PWRDET_PIN, INPUT);
  pinMode(DISTANCE_SENSOR_PIN, INPUT);
  analogReference(EXTERNAL);
}
```

Timer_Task_Setup( ) ⇨ Setup interruption of Timer

```
void Timer_Task_Setup(void){
  //Set Timer3 as a normal timer for LED task
  TCCR3A = 0x00;
  TCCR3B |= _BV(CS32); TCCR3B &= ~_BV(CS31); TCCR3B |= _BV(CS30);
  //Set Timer4 as a normal timer for communcation timeout
  TCCR4A = 0x00;
  TCCR4B |= _BV(CS42); TCCR4B &= ~_BV(CS41); TCCR4B |= _BV(CS40);
  //Set Timer5 as a Fast PWM generator for chest LED driver
  TCCR5A = _BV(COM5A1) | _BV(COM5B1) | _BV(COM5C1) | _BV(WGM51) | _BV(WGM50);
  TCCR5B = _BV(WGM52) | _BV(CS52);
  OCR5A = 0; OCR5B = 0; OCR5C = 0;
}
```

Function

G_SENSOR_Task_Setup( ) ⇒ G sensor initial setting

```
void G_SENSOR_Task_Setup(void){
  Wire.begin();
  setReg(0x2D, 0xA);
}
void setReg(int reg, int data){
    Wire.beginTransmission(I2C_Address);
    Wire.write(reg);
    Wire.write(data);
    Wire.endTransmission();
}
int getData(int reg){
  static int Gsensor_timer;
  Gsensor_timer = 0;
  Wire.beginTransmission(I2C_Address);
  Wire.write(reg);
  Wire.endTransmission();
  Wire.requestFrom(I2C_Address,1);
  if(Wire.available() <= 1 ) return Wire.read();
}
```

Falling_Task( ) ⇒ Check robot is standing or not

```
int Falling_Task(void){
  int posture_index;
  ax = ((getData(0x33) << 8)  + getData(0x32)) / 256.0;
  ay = ((getData(0x35) << 8)  + getData(0x34)) / 256.0;
  az = ((getData(0x37) << 8)  + getData(0x36)) / 256.0;

  if ((az) < -0.75) posture_index=1; //Frontside Getup
  else if ((az) > 0.75) posture_index=2; // Backside Getup
  else if ((ax) < -0.75) posture_index=3; // Rightside Getup
  else if ((ax) > 0.75) posture_index=4; // Leftside Getup
  else if ((az) <= 0.75 && (az) >= -0.75) posture_index=5;// Stand Status

  return posture_index;
}
```

Detected value transfer to G

Function

Getup_Task( ) ⟹ Play get up Sequence

```
void Getup_Task(int posture_index){
 if (posture_index ==1) Action(52);
 else if(posture_index ==2) Action(53);
 else if(posture_index ==3) Action(54);
 else if(posture_index ==4) Action(54);
}
```

IR_SENSOR_Task( ) ⟹ Feedback IR sensor detected value

```
int IR_SENSOR_Task(void){
 distance = (6787/(analogRead(DISTANCE_SENSOR_PIN)-3))-4;
 return distance;
}
```

Detected value transfer to cm

Initial_Pose_Setup( ) ⟹ Play initial Sequence

```
void Initial_Pose_Setup(void){
  XYZrobot.readPose();
  XYZrobot.playSeq(DefaultInitial);
  while(XYZrobot.playing) XYZrobot.play();
}
```

Function

Action( ) ⇨ Play

```
void Action(int N){
  if(N == 0) MusicPlaying_wav_play("none");
  else if(N == 1) {MusicPlaying_wav_play(Music_1);if(ActionNo_1 != None) XYZrobot.playSeq(ActionNo_1);}
  else if(N == 2) {MusicPlaying_wav_play(Music_2);if(ActionNo_2 != None) XYZrobot.playSeq(ActionNo_2);}
  else if(N == 3) {MusicPlaying_wav_play(Music_3);if(ActionNo_3 != None) XYZrobot.playSeq(ActionNo_3);}
  else if(N == 4) {MusicPlaying_wav_play(Music_4);if(ActionNo_4 != None) XYZrobot.playSeq(ActionNo_4);}
  else if(N == 5) {MusicPlaying_wav_play(Music_5);if(ActionNo_5 != None) XYZrobot.playSeq(ActionNo_5);}
  else if(N == 6) {MusicPlaying_wav_play(Music_6);if(ActionNo_6 != None) XYZrobot.playSeq(ActionNo_6);}
  else if(N == 7) {MusicPlaying_wav_play(Music_7);if(ActionNo_7 != None) XYZrobot.playSeq(ActionNo_7);}
  else if(N == 8) {MusicPlaying_wav_play(Music_8);if(ActionNo_8 != None) XYZrobot.playSeq(ActionNo_8);}
  else if(N == 9) {MusicPlaying_wav_play(Music_9);if(ActionNo_9 != None) XYZrobot.playSeq(ActionNo_9);}
  else if(N == 10) {MusicPlaying_wav_play(Music_10);if(ActionNo_10 != None) XYZrobot.playSeq(ActionNo_10);}
  else if(N == 11) {MusicPlaying_wav_play(Music_11);if(ActionNo_11 != None) XYZrobot.playSeq(ActionNo_11);}
  else if(N == 12) {MusicPlaying_wav_play(Music_12);if(ActionNo_12 != None) XYZrobot.playSeq(ActionNo_12);}
  else if(N == 13) {MusicPlaying_wav_play(Music_13);if(ActionNo_13 != None) XYZrobot.playSeq(ActionNo_13);}
  else if(N == 14) {MusicPlaying_wav_play(Music_14);if(ActionNo_14 != None) XYZrobot.playSeq(ActionNo_14);}
  else if(N == 15) {MusicPlaying_wav_play(Music_15);if(ActionNo_15 != None) XYZrobot.playSeq(ActionNo_15);}
```

```
  while((XYZrobot.playing) && !(BT_Packet_Task())){
    XYZrobot.play();
    if(Serial2.available()>0){
      if(BT_Packet_Task()){
        cb_BT();
        break;
      }
      else{
      joystick_status[0] = packet[1];
      joystick_status[1] = packet[2];
      joystick_status[2] = packet[3];
      joystick_status[3] = packet[4];
      }
    }
  }
  if(torque_release){
    A1_16_TorqueOff(A1_16_Broadcast_ID);
    MusicPlaying_wav_stop();
    torque_release = false;
  }
}
```

Read joystick status

Function

BT_Packet_Task( ) ⟹ Read BT packet and torque emergency off judgment.

```
boolean BT_Packet_Task(void){
  //return torque_relase button status
  static int temp_packet[7] = {0};
  static char _i = 0;
  if(Serial2.available() >= 7){
    if((temp_packet[0] = Serial2.read()) == 0) ; else {find_header_BT(); return false;}
    if((temp_packet[1] = Serial2.read()) == 0) {find_header_BT(); return false;}
    if((temp_packet[2] = Serial2.read()) == 0) {find_header_BT(); return false;}
    if((temp_packet[3] = Serial2.read()) == 0) {find_header_BT(); return false;}
    if((temp_packet[4] = Serial2.read()) == 0) {find_header_BT(); return false;}
    if((temp_packet[5] = Serial2.read()) == 0) {find_header_BT(); return false;}
    if((temp_packet[6] = Serial2.read()) == 0) {find_header_BT(); return false;}
    if(temp_packet[1] != 255 && temp_packet[2] != 1){
      Serial2.write((temp_packet[6]&0x00F0)>>4);
    }

    for(_i = 0;_i < 7 ;_i++) packet[_i] = temp_packet[_i];
    BT_update = true;
    if((packet[1]!= 255 && packet[2]!=1)&&((packet[5]&0x0010)>>3)) {
      torque_release = true;
      return true;
    }
    else if(packet[1]==255 && packet[2]==1 && packet[3]==102) {
      torque_release = true;
      return true;
    }
    else{
      torque_release = false;
      return false;
    }
  }
  return false;
}
```

RCU Release button pressed or not

APP Release button pressed or not

Function

BT_Gsensor_Data( ) ⟹ **Transmitting G sensor data in BT packet.**

```
// BT G-sensor Data Feedback
void BT_Gsensor_Data(void){
  g_packet[0]=0;
  if(getData(0x32) == 0xFF){
    g_packet[1] = 0xFF;
    g_packet[7] = 0xC0;
  }
  else{
    g_packet[1] = getData(0x32)+(0x01);
    g_packet[7] = 0x80;
  }
  if(getData(0x33) == 0xFF){
    g_packet[2] = 0xFF;
    g_packet[7] = g_packet[7]+(0x20);
  }
  else{
    g_packet[2] = getData(0x33)+(0x01);
  }
  if(getData(0x34) == 0xFF){
    g_packet[3] = 0xFF;
    g_packet[7] = g_packet[7]+(0x10);
  }
  else{
    g_packet[3] = getData(0x34)+(0x01);
  }
  if(getData(0x35) == 0xFF){
    g_packet[4] = 0xFF;
    g_packet[7] = g_packet[7]+(0x08);
  }
  else{
    g_packet[4] = getData(0x35)+(0x01);
  }
  if(getData(0x36) == 0xFF){
    g_packet[5] = 0xFF;
    g_packet[7] = g_packet[7]+(0x04);
  }
  else{
    g_packet[5] = getData(0x36)+(0x01);
  }
  if(getData(0x37) == 0xFF){
    g_packet[6] = 0xFF;
    g_packet[7] = g_packet[7]+(0x02);
  }
  else{
    g_packet[6] = getData(0x37)+(0x01);
  }

  Serial2.write(g_packet[0]); // packet head
  delay(50);
  Serial2.write(g_packet[1]); // AX_MSB +1
  delay(50);
  Serial2.write(g_packet[2]); // AX_LSB +1
  delay(50);
  Serial2.write(g_packet[3]); // AY_MSB +1
  delay(50);
  Serial2.write(g_packet[4]); // AY_LSB +1
  delay(50);
  Serial2.write(g_packet[5]); // AZ_MSB +1
  delay(50);
  Serial2.write(g_packet[6]); // AZ_LSB +1
  delay(50);
  Serial2.write(g_packet[7]);
  delay(50);
}
```

Transmitting delay time 50ms is much stable in IOS and Android.

Function

BT_IR_Data( ) ⟹ Transmitting IR sensor data in BT packet.

```
void BT_IR_Data(void){
  ir_rowdata = analogRead(DISTANCE_SENSOR_PIN);
  ir_msb = ir_rowdata>>8;
  ir_lsb = ir_rowdata&0xFF;
  ir_packet[0] = 0;
  if(ir_lsb == 0xFF){
    ir_lsb = 0xFF;
    ir_packet[3] = 0x81;
  }
  else{
    ir_lsb = ir_lsb + 0x01;
    ir_packet[3] = 0x80;
  }
  ir_packet[1] = ir_msb +0x01;
  ir_packet[2] = ir_lsb;
  Serial2.write(ir_packet[0]); // packet head
  delay(50);
  Serial2.write(ir_packet[1]); // IR_MSB+1
  delay(50);
  Serial2.write(ir_packet[2]); // IR_LSB+1
  delay(50);
  Serial2.write(ir_packet[3]);
  delay(50);
```

BT_FW( ) ⟹ Feedback firmware version in BT packet.

```
void BT_FW(){
  Serial2.write(0xFF);                    // packet head
  delay(50);
  Serial2.write(model_Bolide);            //Model
  delay(50);
  Serial2.write(type_Y01);                //Type
  delay(50);
  Serial2.write(application_default);     //Application
  delay(50);
  Serial2.write(main_version_number);     //Main Version
  delay(50);
  Serial2.write(secondary_version_number); //Secondary Version
  delay(50);
  Serial2.write(revision_number);         //Revison
  delay(50);
}
```

Function

MusicPlaying_wav_play( ) ⟹ Play Audio，File name range from 0000~9999

```
void MusicPlaying_wav_play(char song_name[]){
  Serial3.write(0);
  Serial3.print("P");
  Serial3.write(song_name); //set the filename of song : 0000 ~ 9999
}
```

MusicPlaying_wav_stop( ) ⟹ Audio off

```
void MusicPlaying_wav_stop(){
  Serial3.write(0);
  Serial3.print("S0000");
}
```

MusicPlaying_wav_volume( ) ⟹ Adjust Audio volume，
value:0x01~0x7F

```
void MusicPlaying_wav_volume(int volume){
  Serial3.write(0);
  Serial3.write('V');
  Serial3.write(volume);// volume : 0x01 ~ 0x7F
  Serial3.print("000");
}
```

Start_Music( ) ⟹ **Power up music**

```
void Start_Music(void){
  int _i = 0x00;
  for(_i = 0 ; _i < 7; _i++){
    tone(BUZZER_PIN, pgm_read_word_near(&start_music_frq[_i]));
    delay(250);
    noTone(BUZZER_PIN);
  }
}
```

Function

Button_Task( ) ⟹ Button pressed setting

```
void BUTTON_Task(void){
  static unsigned char button_timer = 0x00;
  static int key = 0x00, last_key = 0x00;
  key = !digitalRead(BUTTON1_PIN) + ((!digitalRead(BUTTON2_PIN))<<1) + ((!digitalRead(BUTTON3_PIN))<<2) + ((!digitalRead(BUTTON4_PIN))<<3);
  if(key != last_key) button_timer++;
  else button_timer = 0;
  if(button_timer > 20){
    button_timer = 0;
    last_key = key;
    if(key != 0){
      LED_Task(2);
      if(last_key == key_mask_button1) Action(RB_1);
      else if(last_key == key_mask_button2)  Action(RB_2);
      else if(last_key == key_mask_button3)  Action(RB_3);
      else if(last_key == key_mask_button4)  Action(RB_4);
      LED_Task(0);
    }
  }
}
```

Power_Detection_Task( ) ⟹ Low power detection

```
void Power_Detection_Task(void){
  static double PWR_Voltage;
  PWR_Voltage = analogRead(PWRDET_PIN)*0.0124;
  if(PWR_Voltage < Power_Voltage_Alarm) tone(BUZZER_PIN,1000);
}
ISR(TIMER4_OVF_vect){
  Power_Detection_Task();
  packet_timeout_status = true;
  _reset_timer4(timeout_limit);
}
```

Function

LED_Task( ) ⟹ LED mode

```
void LED_Task(char mode){
  if(mode != 0){TCNT3 = -1; _enable_timer3(); LED_mode = mode;}
  else{EYE_LEE_OFF; _disable_timer3(); LED_mode = 0;}
}
ISR(TIMER3_OVF_vect){
  static int R = 0, G = 0, B = 0;
  static int _R = 41, _G = 41, _B = 41;
  static boolean blink_LED = true;
  if (LED_mode == 1){
    if(blink_LED) EYE_LED_BLE;
    else EYE_LED_GRN;
    blink_LED = !blink_LED;
    _reset_timer3(4500);
  }
  else if(LED_mode == 2){
    if(R < 40){R++; OCR5A = pgm_read_word_near(&log_light_40[R]);}
    else if(_R > 0){_R--; OCR5A = pgm_read_word_near(&log_light_40[_R]);}
    else if(G < 40){G++; OCR5B = pgm_read_word_near(&log_light_40[G]); EYE_LED_BLE;}
    else if(_G > 0){_G--; OCR5B = pgm_read_word_near(&log_light_40[_G]);}
    else if(B < 40){B++; OCR5C = pgm_read_word_near(&log_light_40[B]); EYE_LED_GRN;}
    else if(_B > 0){_B--; OCR5C = pgm_read_word_near(&log_light_40[_B]);}
    else{
      R = 0;G = 0;B = 0;
      _R = 41;_G = 41;_B = 41;
    }
    _reset_timer3(200);
    }
    else if (LED_mode == 3){
      if(R < 40){R++; OCR5A = pgm_read_word_near(&log_light_40[R]);}
      else if(_R > 0){_R--; OCR5A = pgm_read_word_near(&log_light_40[_R]);}
      else if(G < 40){G++; OCR5B = pgm_read_word_near(&log_light_40[G]);}
      else if(_G > 0){_G--; OCR5B = pgm_read_word_near(&log_light_40[_G]);}
      else if(B < 40){B++; OCR5C = pgm_read_word_near(&log_light_40[B]);}
      else if(_B > 0){_B--; OCR5C = pgm_read_word_near(&log_light_40[_B]);}
      else{
        R = 0;G = 0;B = 0;
        _R = 41;_G = 41;_B = 41;
      }
      _reset_timer3(200);
    }
}
```

Turn LED off LED_Task(0)

Turn Eyes LED on
LED_Task(1)

Turn both Eyes LED & chest LED on
LED_Task(2)

Turn chest LED on
LED_Task(3)

Y-01_USER_MOTION.h

Official Version ⇨ With G sensor and IR sensor function

```
#ifndef Y-01_USER_MOTION_H
#define Y-01_USER_MOTION_H

#include <avr/pgmspace.h>

#define Adjustment_index true
#define Avoidance_index true
```

Adjustment_index：true, enable G sensor function；false, disable G sensor function

Avoidance_index：true, enable IR sensor function ；
false, disable IR sensor function

* Y-01_USER_MOTION.h which export from XYZrobot Editor ,
both Adjustment_index and Avoidance_index default are false， Sensors are disabled。

Language reference (Structure)

setup()

- The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

Language reference (Structure)

loop()

- After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

Language reference (Structure)

if / else

- if/else allows greater control over the flow of code than the basic if statement, by allowing multiple tests to be grouped together. For example, an analog input could be tested and one action taken if the input was less than 500, and another action taken if the input was 500 or greater.

Language reference (Structure)

- **else can proceed another if test, so that multiple, mutually exclusive tests can be run at the same time.**

- ```
  if (pinFiveInput < 500)
   {
     // do Thing A
    }
  else if (pinFiveInput >= 1000)
  {
     // do Thing B
   }
  else { // do Thing C }
  ```
  ```

Language reference (Structure)

do - while

- The do loop works in the same manner as the while loop, with the exception that the condition is tested at the end of the loop, so the do loop will always run at least once.

Language reference (Structure)

#Define

- ***#define*** is a useful C component that allows the programmer to give a name to a constant value before the program is compiled. Defined constants in arduino don't take up any program memory space on the chip. The compiler will replace references to these constants with the defined value at compile time.
- This can have some unwanted side effects though, if for example, a constant name that had been #defined is included in some other constant or variable name. In that case the text would be replaced by the #defined number (or text).
- In general, the const keyword is preferred for defining constants and should be used instead of #define.
- Arduino defines have the same syntax as C defines:

  Syntax

```
#define constantName value
```

Language reference (Structure)

#include

- ***#include*** **is used to include outside libraries in your sketch. This gives the programmer access to a large group of standard C libraries (groups of pre-made functions), and also libraries written especially for Arduino.**
- **Note that #include, similar to #define, has no semicolon terminator, and the compiler will yield cryptic error messages if you add one.**

Example

This example includes a library that is used to put data into the program space *flash* instead of *ram*. This saves the ram space for dynamic memory needs and makes large lookup tables more practical.

```
#include <avr/pgmspace.h>

prog_uint16_t myConstants[] PROGMEM = {0, 21140, 702   , 9128,  0, 25764, 8456,
0,0,0,0,0,0,0,0,29810,8968,29762,29762,4500};
```

Language reference (Structure)

Comparison Operators

- == (equal to)
- != (not equal to)
- < (less than)
- > (greater than)
- <= (less than or equal to)
- >= (greater than or equal to)

Boolean Operators

- && (and)
- || (or)
- ! (not)

Language reference (Variables)

INPUT

constants

- Constants are predefined expressions in the Arduino language. They are used to make the programs easier to read. We classify constants in groups:

  Defining Logical Levels: true and false (Boolean Constants)

- There are two constants used to represent truth and falsity in the Arduino language: true, and false.
- false is the easier of the two to define. false is defined as 0 (zero).
- true is often said to be defined as 1, which is correct, but true has a wider definition. Any integer which is non-zero is true, in a Boolean sense. So -1, 2 and -200 are all defined as true, too, in a Boolean sense.
- Note that the true and false constants are typed in lowercase unlike HIGH, LOW, INPUT, and OUTPUT.

Language reference (Variables)

OUTPUT

constants

- Constants are predefined expressions in the Arduino language. They are used to make the programs easier to read. We classify constants in groups:

  Defining Logical Levels: true and false (Boolean Constants)

- There are two constants used to represent truth and falsity in the Arduino language: true, and false.
- false is the easier of the two to define. false is defined as 0 (zero).
- true is often said to be defined as 1, which is correct, but true has a wider definition. Any integer which is non-zero is true, in a Boolean sense. So -1, 2 and -200 are all defined as true, too, in a Boolean sense.
- Note that the true and false constants are typed in lowercase unlike HIGH, LOW, INPUT, and OUTPUT.

Language reference (Variables)

float

Description

- Data type for floating-point numbers, a number that has a decimal point. Floating-point numbers are often used to approximate analog and continuous values because they have greater resolution than integers. Floating-point numbers can be as large as 3.4028235E+38 and as low as -3.4028235E+38. They are stored as 32 bits (4 bytes) of information.
- Floats have only 6-7 decimal digits of precision. That means the total number of digits, not the number to the right of the decimal point. Unlike other platforms, where you can get more precision by using a double (e.g. up to 15 digits), on the Arduino, double is the same size as float
- Floating point numbers are not exact, and may yield strange results when compared. For example 6.0 / 3.0 may not equal 2.0. You should instead check that the absolute value of the difference between the numbers is less than some small number.

Language reference (Variables)

- Floating point math is also much slower than integer math in performing calculations, so should be avoided if, for example, a loop has to run at top speed for a critical timing function. Programmers often go to some lengths to convert floating point calculations to integer math to increase speed.
- If doing math with floats, you need to add a decimal point, otherwise it will be treated as an int.

Example Code

```
int x;
int y;
float z;

x = 1;
y = x / 2;            // y now contains 0, ints can't hold fractions
z = (float)x / 2.0;   // z now contains .5 (you have to use 2.0, not 2)
```

Language reference (Variables)

PROGMEM

- Store data in flash (program) memory instead of SRAM.
- The PROGMEM keyword is a variable modifier, it should be used only with the datatypes defined in *Mask_Definition.h*. It tells the compiler "put this information into flash memory", instead of into SRAM, where it would normally go.
- PROGMEM is part of the *Mask_Definition.h* library that is available in the AVR architecture only. So you first need to include the library at the top your sketch, like this:

```
#include "Mask_Definition.h"
```

Syntax

- dataType variableName[] PROGMEM = {data0, data1, data3...};

Memory

- There are three pools of memory in the microcontrollers (e.g. theATmega168):

1.Flash memory (program space), is where the Arduino sketch is stored.

2.SRAM (static random access memory) is where the sketch creates and manipulates variables when it runs.

3.EEPROM is memory space that programmers can use to store long-term information.

- Flash memory and EEPROM can't chamemory are non-volatile (the information persists after the power is turned off). SRAM is volatile and will be lost when the power is cycled.

Note: Flash (PROGMEM) memory can only be populated at program burn time. You nge the values in the flash after the program has started running.

- The amounts of memory for various microcontrollers used on boards are as follows:

	ATMega168	ATMega328P	ATmega1280	ATmega2560
Flash (1 Kbyte used for bootloader)	16 KBytes	32 KBytes	128 KBytes	256 KBytes
SRAM	1024 bytes	2048 bytes	8 KBytes	8 KBytes
EEPROM	512 bytes	1024 bytes	4 KBytes	4 KBytes

Thank You!