Introduction to Mechatronic

Iran University of Science and Technology

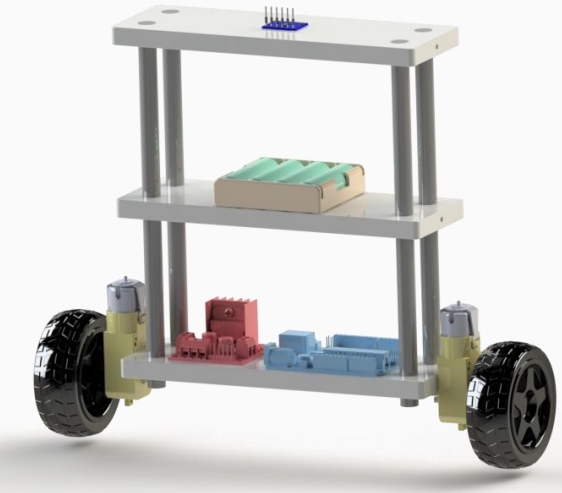
Mini segway

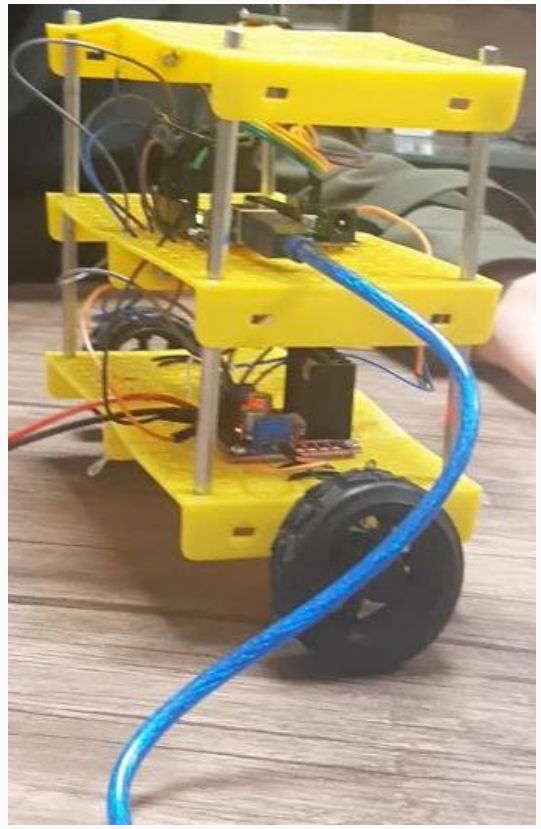
Self-balance Robot

Context

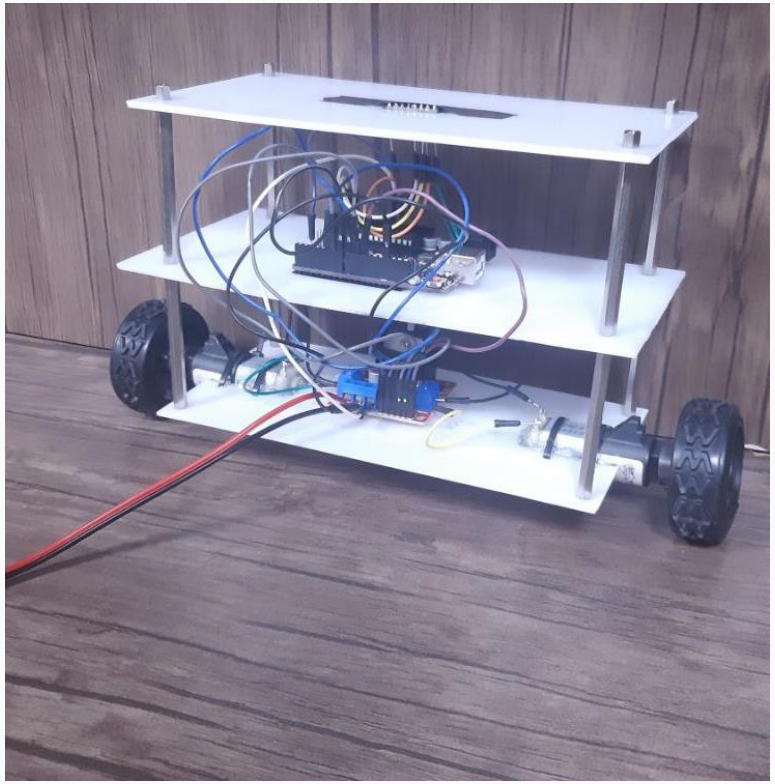
1. First Vs. Final Figuration
2. Components
3. Challenges & solutions
4. Final Code with illustrations

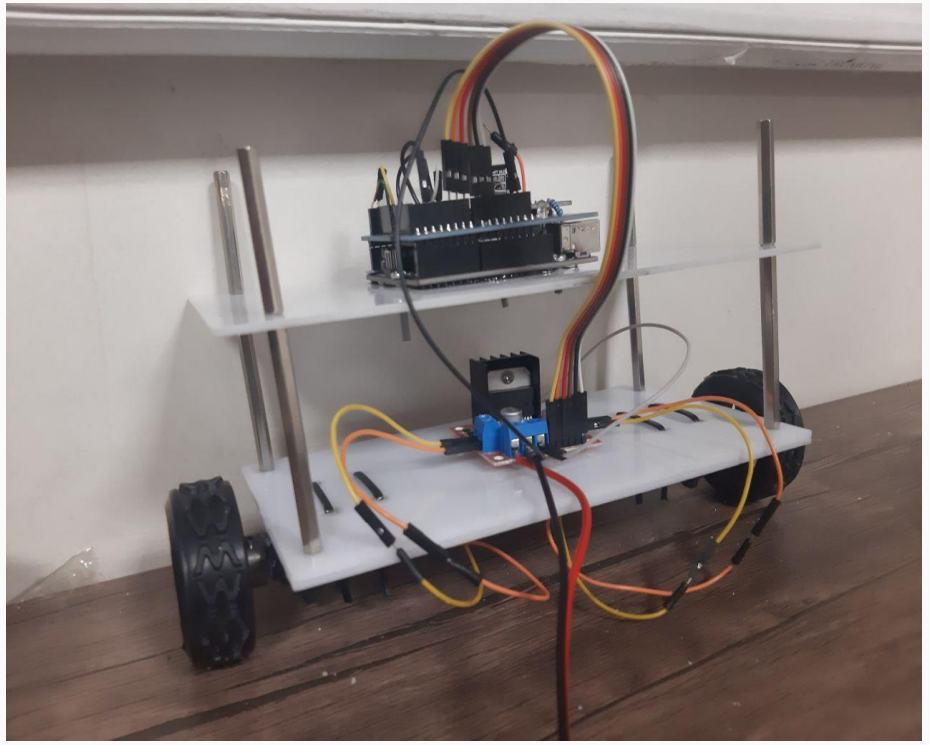
1 First Vs. Final Figuration



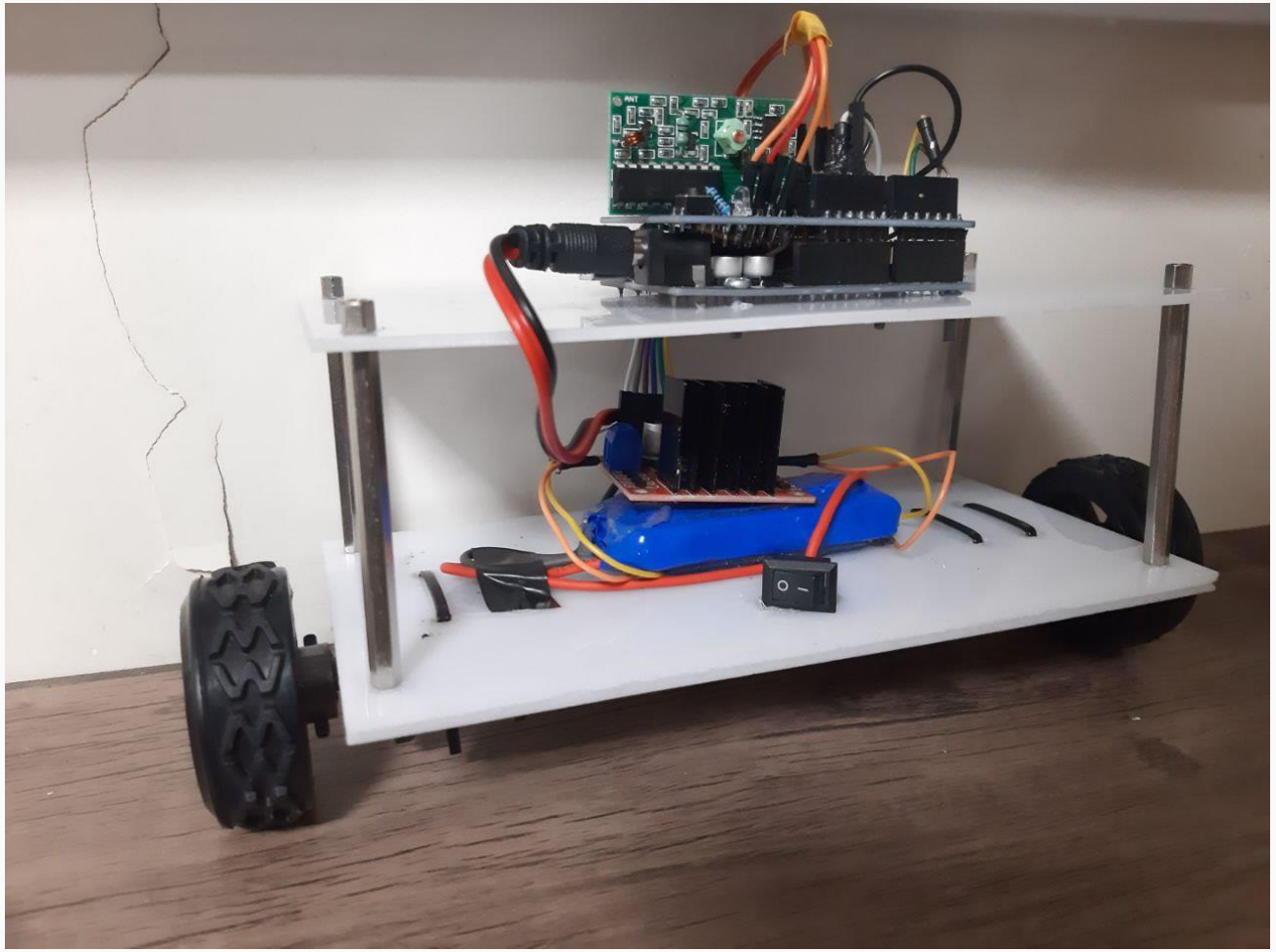


1 First Vs. Final Figuration





1 First Vs. Final Figuration



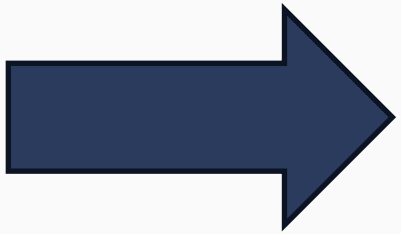
3 Challenges & solutions

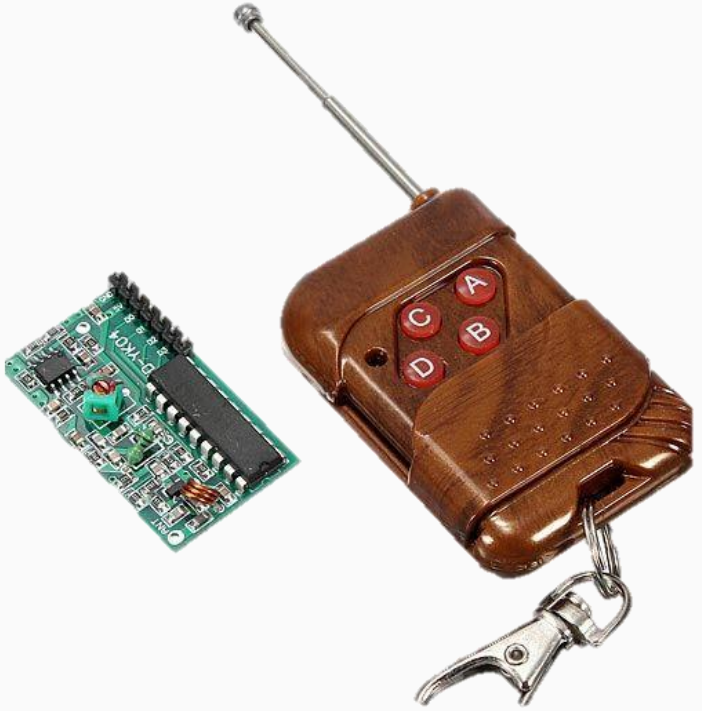
First： Noise & errors while capturing data

- The size and the materials of the plate
- No-linear effects on the Pitch angle
- Long distance between the sensor and the Arduino

3 Challenges & solutions

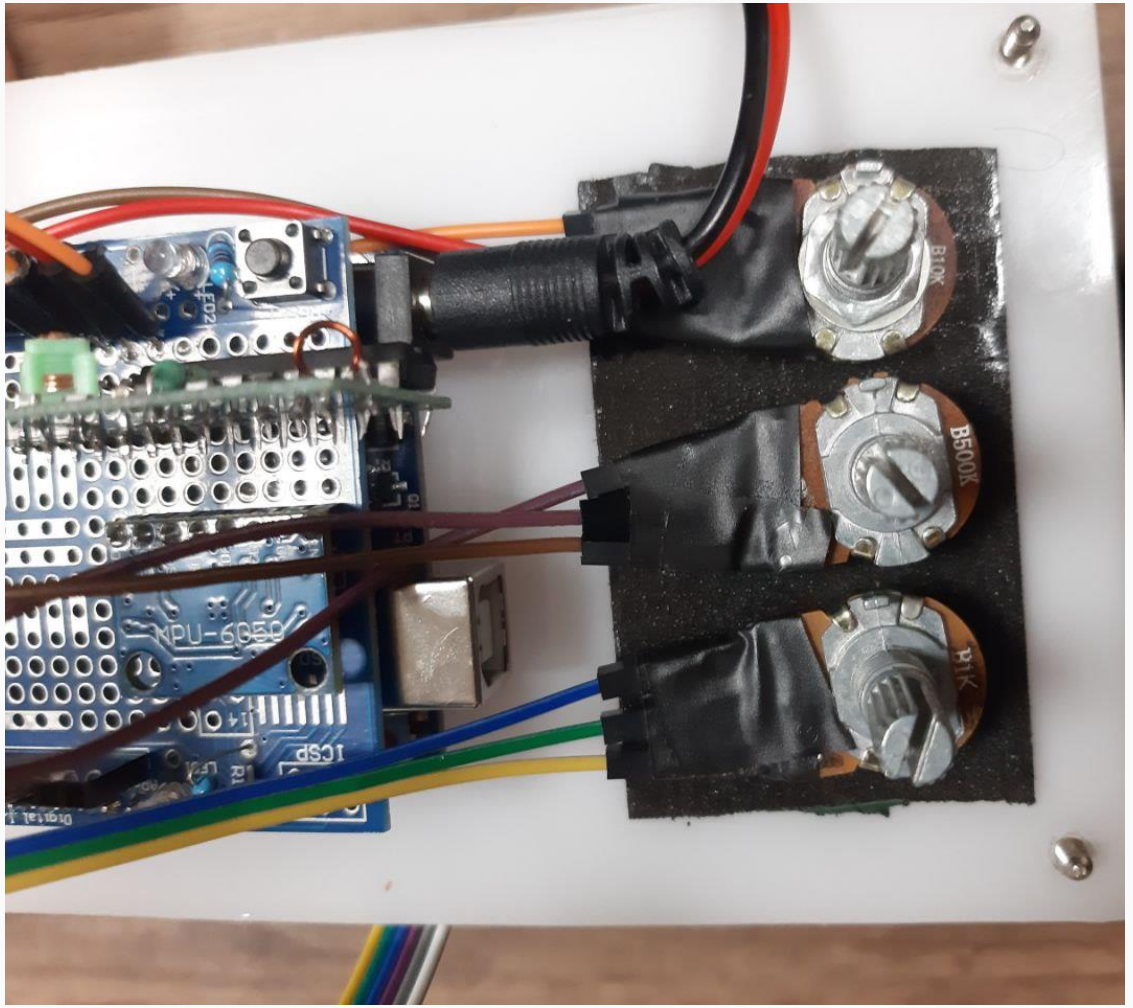
Second: Finding the best remote





3 Challenges & solutions

3rd: Tuning PID parameters using potentiometers



4 Final Code with illustrations

1st: Importing Libs & Defining variables

```
#include <PID_v1.h>

#include <PID_v1.h>
#include <LMotorController.h>
#include <I2Cdev.h>
#include <MPU6050_6Axis_MotionApps20.h>

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
 #include "Wire.h"
#endif

#define MIN_ABS_SPEED 237

MPU6050 mpu;
```

```
// MPU control/status vars
bool dmpReady = false; // set true if DMP init was successful
uint8_t mpuIntStatus; // holds actual interrupt status byte from MPU
uint8_t devStatus; // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize; // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount; // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q; // [w, x, y, z] quaternion container
VectorFloat gravity; // [x, y, z] gravity vector
float ypr[3]; // [yaw, pitch, roll] yaw/pitch/roll container and gravity vector

```

4 Final Code with illustrations

2nd : Defining Variables

```
29  //PID
30  double originalSetpoint = 172.5;
31  double setpoint = originalSetpoint;
32  double movingAngleOffset = 0.1;
33  double input, output;
34
35  //adjustthese values to fit your own design
36  double Kp =0.9;
37  double Kd = 0.001;
38  double Ki = 5;
39  PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);
40
41  double motorSpeedFactorLeft = 0.7;
42  double motorSpeedFactorRight = 0.65;
43
```

4 Final Code with illustrations

3rd : Controller's Variables

```
44  //MOTOR CONTROLLER
45  int ENA = 5;
46  int IN1 = 6;
47  int IN2 = 7;
48  int IN3 = 9;
49  int IN4 = 8;
50  int ENB = 10;
51  LMotorController motorController(ENA, IN1, IN2, ENB, IN3, IN4, motorSpeedFactorLeft, motorSpeedFactorRight);
52
53  volatile bool mpuInterrupt = false; // indicates whether MPU interrupt pin has gone high
54  void dmpDataReady()
55  {
56   mpuInterrupt = true;
57  }
58
59   //remote
60   int ApinState = digitalRead(13);
61   int BpinState = digitalRead(3);
62   int CpinState = digitalRead(12);
63   int DpinState = digitalRead(4);
64
```

4 Final Code with illustrations

4th : Setup Fn

```
66  void setup()
67  {
68   Serial.begin(9600);
69   pinMode(13, INPUT);
70   // join I2C bus (I2Cdev library doesn't do this automatically)
71
72   #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
73   Wire.begin();
74   TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
75   #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
76   Fastwire::setup(400, true);
77   #endif
78
79   mpu.initialize();
80
81   devStatus = mpu.dmpInitialize();
82
83  // supply your own gyro offsets here, scaled for min sensitivity
84  mpu.setXGyroOffset(-541.08);
85  mpu.setYGyroOffset(10.78);
86  mpu.setZGyroOffset(-103.23);
87  mpu.setZAccelOffset(-14998.52); // 1688 factory default for my test chip
88
89  // make sure it worked (returns 0 if so)
90  if (devStatus == 0)
91  {
92  // turn on the DMP, now that it's ready
93  mpu.setDMPEnabled(true);
94
95  // enable Arduino interrupt detection
96  attachInterrupt(0, dmpDataReady, RISING);
97  mpuIntStatus = mpu.getIntStatus();
98
```

4 Final Code with illustrations

5th : Setup Fn

```
99    // set our DMP Ready flag so the main loop() function knows it's okay to use it
100   dmpReady = true;
101
102   // get expected DMP packet size for later comparison
103   packetSize = mpu.dmpGetFIFOPacketSize();
104
105   //setup PID
106   pid.SetMode(AUTOMATIC);
107   pid.SetSampleTime(10);
108   pid.SetOutputLimits(-255, 255);
109   }
110   else
111   {
112   // ERROR!
113   // 1 = initial memory load failed
114   // 2 = DMP configuration updates failed
115   // (if it's going to break, usually the code will be 1)
116   Serial.print(F("DMP Initialization failed (code "));
117   Serial.print(devStatus);
118   Serial.println(F(")"));
119   }
120  delay(15);
121  }
```

4 Final Code with illustrations

6th : PID Controller

```
123
124   void loop()
125   {
126    // if programming failed, don't try to do anything
127    if (!dmpReady) return;
128
129    // wait for MPU interrupt or extra packet(s) available
130   while (!mpuInterrupt && fifoCount < packetSize)
131
132    {
133    //no mpu data - performing PID calculations and output to motors
134
135    //int pot1=analogRead(A0);
136    //Kp =map(pot1,0,1023,0,350);
137    //Serial.println(Kp);
138    pid.Compute();
139    //Serial.println(output);
140    //motorController.move(output, MIN_ABS_SPEED);
```

4 Final Code with illustrations

Tuning PID params manually

$$\begin{cases} KP \begin{cases} LOW: The\ response\ is\ not\ strong\ enough \\ HIGH: Oscilatory\ motion\ around\ the\ setpoint \end{cases} \\ KI: \begin{cases} LOW: The\ respose\ is\ far\ slow(with\ respect\ to\ deviations) \\ HIGH: Overshoot\ surging \end{cases} \\ KD: \{LOW: Sedate\ the\ respose \end{cases}$$

4

Final Code with illustrations

7th : Remote Controller & Command

```
if(ApinState == HIGH) {
 double Kp =0.689;
 double Kd = 0.009;
 double Ki = 5;
  motorController.move(output, MIN_ABS_SPEED); //Moving Forward

}
else if(CpinState==HIGH){
 double Kp =0.689;
 double Kd = 0.009;
 double Ki = 5;
  double motorSpeedFactorLeft = 0.8;
  double motorSpeedFactorRight = 0.3;
  motorController.move(output, MIN_ABS_SPEED); //CCW Rotation
}
else if (BpinState==HIGH){

}
else if (DpinState==HIGH){
}
else {
  motorController.move(output, 0); //Stand
}
}
```

4 Final Code with illustrations

8th: Command with "LMotorController" library

```
void LMotorController::move(int speed, int minAbsSpeed)
{
    int direction = 1;

    if (speed < 0)
    {
        // printf("speed -\n");
        direction = -1;

        speed = min(speed, -1*minAbsSpeed);
        speed = max(speed, -255);
    }
    else
    {
        // printf("speed +\n");
        speed = max(speed, minAbsSpeed);
        speed = min(speed, 255);
    }

    if (speed == _currentSpeed) return;

    int realSpeed = max(minAbsSpeed, abs(speed));

    digitalWrite(_in1, speed > 0 ? HIGH : LOW);
    digitalWrite(_in2, speed > 0 ? LOW : HIGH);
    digitalWrite(_in3, speed > 0 ? HIGH : LOW);
    digitalWrite(_in4, speed > 0 ? LOW : HIGH);
    analogWrite(_ena, realSpeed * _motorAConst);
    analogWrite(_enb, realSpeed * _motorBConst);

    _currentSpeed = direction * realSpeed;
}
```

4 Final Code with illustrations

9th: Check sensor connections and data capturing

```
// reset interrupt flag and get INT_STATUS byte
mpuInterrupt = false;
mpuIntStatus = mpu.getIntStatus();

// get current FIFO count
fifoCount = mpu.getFIFOCount();

// check for overflow (this should never happen unless our code is too inefficient)
if ((mpuIntStatus & 0x10) || fifoCount == 1024)
{
// reset so we can continue cleanly
mpu.resetFIFO();
Serial.println(F("FIFO overflow!"));

// otherwise, check for DMP data ready interrupt (this should happen frequently)
}
else if (mpuIntStatus & 0x02)
{
// wait for correct available data length, should be a VERY short wait
while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

// read a packet from FIFO
mpu.getFIFOBytes(fifoBuffer, packetSize);
```

4 Final Code with illustrations

10th : Receiving data from the Pith angle(PID INPUT)

```
// track FIFO count here in case there is > 1 packet available
// (this lets us immediately read more without waiting for an interrupt)
fifoCount -= packetSize;

mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
input = ypr[1] * 180/M_PI + 180;
Serial.println(output);
```

RESULTS

https://drive.google.com/file/d/13OqDroXF4NBBrQuub_YbH93DICVu7qwH/view?usp=sharing