Tethering System for Unmanned Aerial Vehicles

Quentin Morales-Perryman (Hampton University, Electrical Engineering), *SUNFEST Fellow*

Dr. Daniel D. Lee, Department of Electrical Systems Engineering

***Abstract*— With the use of unmanned aerial vehicles accelerating and government regulation restricting the free flight of these vehicles, a tether is required to continue using unmanned aerial vehicles (UAVs) in outdoor conditions. Unfortunately, a versatile tether is unavailable. Current tethers are simply lines that hook the vehicle to some object on the ground. What was needed was a tether which could respond to the strength at which the UAV at the end is pulling as well as a tether that avoids getting tangled within the UAV itself. We built a tether which is capable of being attached to an unmanned aerial vehicle and a ground object and keeps a constant tension on the line to keep it from tangling within the vehicle itself and that avoids impeding the flight pattern of the vehicle. This report will detail the procedure which we used to develop this tether as well as further developments that can be made to enhance the tether.**

***Index Terms*—UAV tether**

I. INTRODUCTION

Due to regulations set forth by the Federal Aviation Administration, the ability to fly unmanned aerial vehicles (UAVs) in outdoor areas is restricted. Only UAVs that are tethered to the ground or to objects on the ground can be flown outside unless special certification is obtained. This rule does not apply to hobbyists or to those performing experiments with UAVs that are strictly done indoors [1].

The goal of this project was to find or create an adaptable device to connect a UAV to an object on the ground in a way that does not 1) weigh down the UAV, 2) tangle up within the UAV or ground object, and 3) detract from the main goal of the UAV, such as obstruct the view of a camera placed on the UAV?

The original solution was to use a premade tether to connect the UAV to a ground object. The problem was that no tether had, thus far, provided the amount of control and adaptability that is required by the UAVs that are utilizing them. I had taken on the assignment to create a versatile tether to attach a quadrotor to a ground robot.

The creation of a useful tether requires combining both hardware and software components. However, the ability to join these two components together seamlessly is a complex and time consuming task.

The following section will provide an explanation of key topics used throughout this paper. The subsequent sections will explain the procedure that was used to design and develop the tether, as well as provide further potential applications and improvements that can be applied to the tether.

II. BACKGROUND

A. Motivation

As part of a larger project, I developed a tether to connect a quadrotor to a ground robot. Here is a background explanation on what the larger project entails.

In 2010, Australia hosted an event called the Multi Autonomous Ground-robotic Competition (MAGIC 2010). The goal of MAGIC 2010 was for teams of students to create a group of autonomous robots with the ability to explore terrain and execute given missions such as surveying and mapping an unknown environment [2].

In respect to our project, the goal was to build on the foundation developed by MAGIC 2010, which simply opened the door to the possibilities and difficulties that existed with autonomous robots. To do the project, we needed a way for a group of robots to map unknown terrain without the use of an external mapping tool, such as GPS. To solve this problem the simultaneous localization and mapping (SLAM) algorithm was used. This addressed the problem of a robot navigating in an unknown environment [3]. Incorporating the SLAM algorithm into the robots' software gave them the ability to create a map of their surroundings as well as the ability to place themselves in the map.

We also needed a way for the robots to coordinate and communicate information with each other so they could work together autonomously. To meet the goal, we made a team consisting of ground robots paired with quadrotors. The ground units are the processing station where the majority of the computation is done. The quadrotor flies above and implements a tag identification system in order to locate the position of any surrounding ground robots, and sends the information to the paired ground robot for processing. The quadrotors and ground robots are paired together using the tether system we helped to develop.

B. PID Control

A PID control consists of proportional, integral, and derivative terms to provide feedback control of required processes [4]. The proportional term corrects for current errors, the integral term corrects for past errors, and the derivative term corrects for future errors. The ability of the robots to make adjustments to their position, speed, and direction is based in part on PID control. PID control is essential to reduce the amount of miscalculations that naturally occur due to the nature of the programs being run. PID control is modeled after the following equation:

$$Output = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \quad \text{(Eq. 1)}$$

K_p is the proportional gain. This is the constant that contributed the bulk of our change in output. By increasing this proportional gain, current errors have a greater effect on the output. This increase provides faster reaction time to immediate changes in the environment.

K_i is the integral gain. Adjusting this value allows previous error to have more or less effect on the output. Adjusting this value also corrects for the steady state error in our value which is the constant offset from the setpoint (the target value that the system is trying to reach) caused by the proportional term. The proportional term needs this steady state error to occur in order to function and the integral gain helps correct for it.

K_d is the derivative gain. Adjusting this value adjusts how quickly the error returns to its set point value and adds stability to the output. It is used to reduce set point overshoot as well as oscillations around the setpoint.

e(t) is the error at time t.

The Output is the resulting combination of the proportional, integral, and derivative terms and is used to adjust the process inputs for future reference.

C. Ground Robot

Originally developed by the University of Pennsylvania for MAGIC 2010, the robots are being modified to fit the current project undertaken by the students of the University of Pennsylvania. An example of one of the ground robots is depicted in Fig. 1.

Fig. 1 A ground robot originally created by the University of Pennsylvania

D. Unmanned Aerial Vehicle (UAV)

An unmanned aerial vehicle is any type of flying or hovering vehicle that does not have a human pilot on board. This definition includes both autonomous (where the vehicle is simply controlled by a computer) and semi-autonomous (where a pilot on the ground guides the vehicle by remote control) [5].

E. Quad Rotor

A quad rotor is a small flying device propelled by four rotors. A quad rotor uses it rotors to maneuver by adjusting the rotation speed of the rotors to create an unbalanced force [6]. That force then either changes the lift, direction, position, or some combination of them to move in the air. A quadrotor has a compact size, high level of dexterity, and general ease in performance that other aerial vehicles could not obtain. An example of one of the quadrotors currently being used is depicted in Fig. 1.

Fig. 2 Quadrotor created by Kmel Electronics

F. Hall Effect in a Current Sensor

Discovered in 1879 by Edwin H. Hall, the Hall Effect is a measurable transverse force. A transverse force exerted on moving charge carriers when an electric current flows through a conductor in a magnetic field. This force causes the charge carriers to buildup on one side of the conductor and produces a measurable voltage difference between the two sides of the conductor. This voltage is the transverse voltage. The current sensor uses this voltage to get an estimate of the current flowing through the conductor [7].

G. Exponential Filter

The exponential filter applies a weighting factor to a moving average which decreases exponentially with time. The filter has 2 main components: the previous filtered value and the current input. These components are modeled in the equation as follows

$$y(k) = a * y(k-1) + (1-a) * x(k) \quad \text{(Eq. 2)}$$

Where x(k) is the current input at time step k and y(k) is the filtered value at time step k. The variable a is the weighting factor. The value of a is between 0 and 1, although typically between 0.8 and 0.99. The value of *a* can be adjusted up or down depending on how much of an effect you want the most recent term influencing your filtered value. A higher value for *a* will give the most recent term less of an effect on the filtered value, a lower value for *a* will give the most recent term a greater effect on the filtered value [8].

III. Procedure

In order to build the tether, we first made a list of the functions that we needed the tether to possess. These functions were 1) enough tension in the line to prevent entanglement, 2) enough slack in the line so the flight of the motion of the quadrotor is not impeded, and 3) the ability of the tether to continually adjust the tension in the line in order to maintain the ideal tension in the line.

10-lb test fishing line was chosen to use as the tether for this project because it satisfied the necessary criteria, as follows: 1) light weight to reduce strain on quadrotor and reduce the impedance of motion and 2) high strength to withstand constant wear and possible encounters with the elements of nature. In addition the material was easily available and cost effective.

We decided that the most effective method to control the tethering system for the quadrotor was to use a dc motor controlled by a PID controller that received input from a current sensor. Due to the unpredictable flight that might be experienced by the quadrotor, an open loop system that fed information to a PID controller seemed like the best approach to take.

We created a spool to attach to the motor and house the fishing line. The spool was created with acrylonitrile butadiene styrene (ABS) plastic on a 3-D printer. ABS plastic is highly durable and smooth which reduces friction and makes for an ideal line holding material. Because a simple spool design allowed the line to slip, we designed a spool with a v-shape in order to funnel the line to the center of the spool and contain the slipping line. The following picture (Fig. 3) shows the final spool.

Fig. 3 The final spool design

After designing the mount, we needed a way to incorporate PID functionality into the tether. To do this, we gathered several devices which include: a microcontroller, an H-bridge, and a current sensor.

The microcontroller we used was called the Seeeduino Mega v1.1, which was developed by Arduino and is depicted in Fig. 6. The microcontroller is programmed through the Arduino sketch in a combination of C and C++ languages. The microcontroller had the job to take in the analog signal from the current sensor and convert that into a value set within a range dictated by the 10-bit input (so between 0 and 1023). Based on this value, the microcontroller had to be able to run PID-based code and incorporate the equation:

$$V = IR \qquad \text{(Eq. 3)}$$

Where V is the voltage and is directly proportional to the current. I is the current and is the term being affected by the pull of the line on the motor. R is the resistance and is nearly constant on the motor.

As the tension in the line increases or decreases the torque the motor exerts changes. This causes an increase or decrease in the amount of current the motor uses and a proportional increase or decrease of the voltage across the motor. The current sensor takes the current and feeds an equivalent analog signal to the microcontroller.

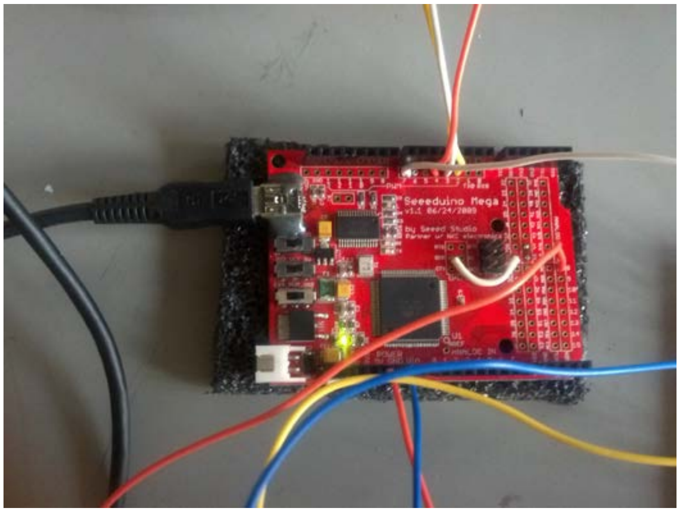

Fig. 4 Seeeduino Mega v1.1 microcontroller developed by Arduino

The PID control program on the microcontroller then decides whether to adjust the value up or down according to the difference between the analog signal and a setpoint value. The difference is the error. The setpoint value was determined by testing done to determine what value was needed to keep a certain amount of tension on the line in order to prevent the line from becoming too loose and tangling in the motor and becoming too tight and impeding the flight of the quad rotor. The microcontroller then outputs an adjusted signal to the motor pin that connects to the H-bridge.

The H-bridge (depicted in Fig. 5) then takes that signal and uses it to increase or decrease the speed of the motor which will bring the current in the motor closer to the predetermined setpoint and keep the line at the right tension.

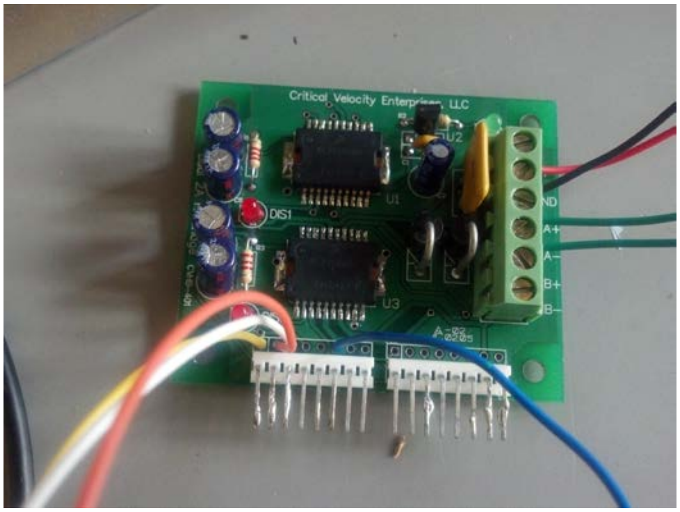

Fig. 5 Dual 2A H-bridge developed by Critical Velocity Enterprises, LLC

The current sensor we used was an ACS712 low current sensor breakout developed by SparkFun Electronics to measure the current flowing through the motor using the Hall Effect. The current sensor is depicted in Fig. 6. The current sensor measures current up to 5 A of either AC or DC current and outputs an analog signal which varies linearly which the current that is sensed. For our purposes, the current sensor was used to sense current that varied from 60 to 130 mA. In order to adjust the current sensor to find our range of current, we had to adjust the reference voltage to the level that corresponded with our current level as well as adjust the gain potentiometer to boost the signal that was being output by the current sensor to a strong enough level that could be used to adjust the motor.

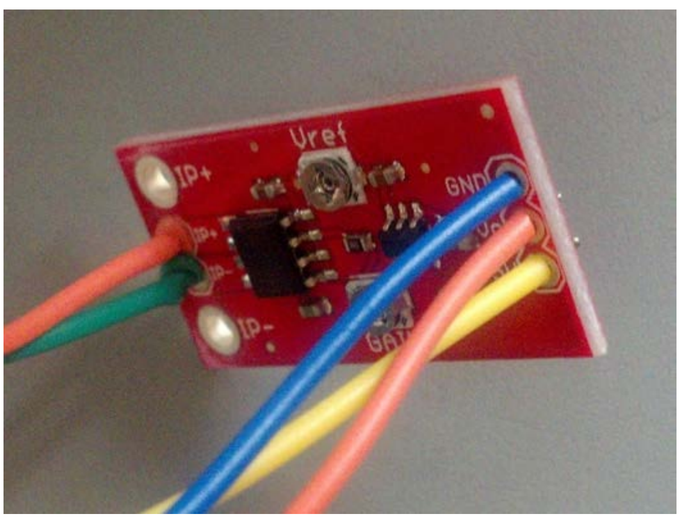

Fig. 6 ACS712 Low Current Sensor by SparkFun Electronics

The current sensor is powered by 5-volt DC power which it draws from the microcontroller. The H-bridge is a Dual 2A H-bridge developed by Critical Velocity Enterprises, LLC. The H-bridge is run off of a 12-volt DC battery supply. After properly connecting all of the components together, the microcontroller had to be properly programmed so as to become a PID controller. The code used to program the microcontroller can be found in the appendix.

The system uses a PID controller with an exponential filter to reduce noise. An exponential filter, which requires no memory, allowed us to increase the effect that the most recent current measurement from the current sensor had on the PID controller which was a notable advantage. After the exponential filter was implemented, the readings from the PID controller smoothed out and became consistent which the current flowing through the motor.

Testing allowed us to determine the correct proportional, integral, and derivative gains to apply to the PID controller to obtain the correct motor control for the tether.

We created a mount to hold the motor, spool, and electronic pieces of the PID controller. The mount was designed using SolidWorks design software and cut out using a laser cutter. The material used was one-quarter inch thick medium-density fiberboard (MDF). The design incorporated the use of press-fit and interlocking pieces in order to give the mount a stable frame.

The mount is approximately 4.5in by 9in by 4in. with two compartments: one housing the motor and the other housing the microcontroller, H-bridge, current sensor, and breadboard. The mount is connected to the ground robot by use of two 12in aluminum rods which are press fit through the circular holes on the mount and screwed onto the ground robot. The mount is depicted in Fig. 7.

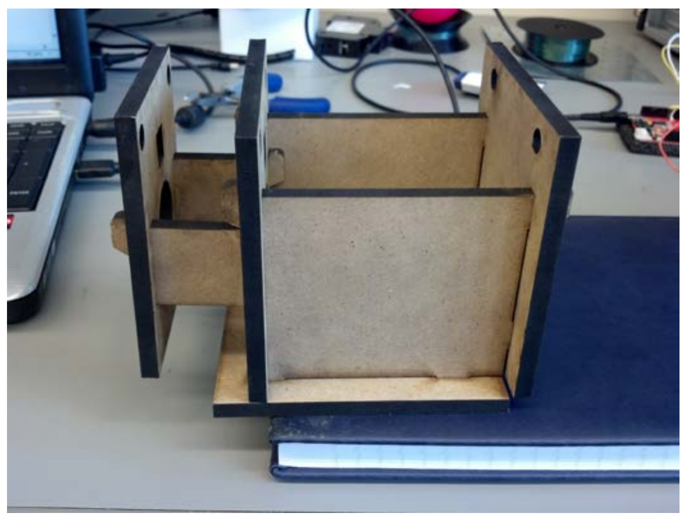

Fig. 7 Final mount design

After attaching the mount to the ground robot, the tether was ready for its final round of testing and calibration. Previously we were testing the tether by horizontally pulling on the line to figure out the ideal tension and rate at which to speed up and slow down the motor. However, when we began to test the tether vertically, which is how it will be used in practice, the tension was too strong. It pulled on the quadrotor and impeded its flight. The problem was simple to solve in that we just had to go back and adjust down the setpoint for the PID controller. This led to a reduced line tension and a functioning tether for the team to use.

IV. Discussion and Conclusion

In developing the tether, many pieces needed to work together but the key was adjustability. The tether has to always adjust to the demands placed on it by the quadrotor. This led us to believe a PID controller would be the best fit based on the ability to respond to changes in a relatively quick manner.

As a possible improvement on this tether design, the use of a power cable instead of fishing line to connect the ground object to the UAV could be used. This change could greatly increase the flight time of the UAV, which in most designs is limited by the UAV's battery life. Additionally, a cord that could transmit data from the UAV to the ground object could be a helpful improvement. This would probably increase the rate at which UAVs could transfer data as well as solve any problems with accessing the data the UAVs collect.

A potential problem that can arise from this with respect to the tether controller is adjusting the controller for the additional weight the data or power cables would have as well as adjusting the size of the spool and the motor speed to compensate for the increase in the cable size that is likely to occur.

Until government regulations are changed, researchers and any other company developing or testing UAVs will have to keep their vehicles tethered to the ground. This will keep tethering systems such as this one in demand for the foreseeable future.

Appendix

Code programmed in microcontroller to control tether

```
/*
 * Simple Program to drive DC Motor
 * (Seeeduino Mega v1.1)
 */

//Variable assigning pin numbers and states
const int motorPin = 3;
const int inPin1 = 2;
const int inPin2 = 4;
const int analogModePin = A8;
const int manualSwitchPin = 7;
int manualOn = LOW;

//Variables to ensure button is supposed to be pushed
long lastDebounce = 0;
int debounceDelay = 50;
int buttonState;
int lastButtonState = LOW;

const int hover = 50; //Signal needed to sustain flight
const int pullIn = 150; //Signal needed to reel in quad

//For PID function
const int idealCurr = 540;
unsigned long totalTime = 0;
long lastError = 0;
int sum = 0;
int error = 0;
int iPart = 0;

void setup()
{
  Serial.begin(9600);
  pinMode(motorPin, OUTPUT);
  pinMode(inPin1, OUTPUT);
  pinMode(inPin2, OUTPUT);
  pinMode(analogModePin, INPUT);

  //draws motor line in
  //switch inPin1 and inPin2 to reverse motor direction
  digitalWrite(inPin1, LOW);
  digitalWrite(inPin2, HIGH);
}

/*
 * Gives feedback to function to make adjustments
 * to line tension
 */
int pidControl()
{
  int input, output, pPart, dPart;
  long deltaT = millis() - totalTime;
  totalTime = millis();

  //exponential filter
  float a = 0.925;
  input = analogRead(analogModePin);
  sum = a * sum + (1 - a) * input;

  error = idealCurr - sum;

  float kp = 19.5;
  pPart = kp * error;

  float ki = 4.15;
  iPart += ki * error;

  //limits integral gain to 25%
  if(iPart > 250)
    iPart = 250;

  if(iPart < -250)
    iPart = -250;

  float kd = 5.15;
  dPart = (kd * (error - lastError)) / deltaT;

  lastError = error;
  output = pPart + iPart + dPart;

  //limits total error gain
```

```
  if(output < -1000)
    output = -1000;

  if(output > 1000)
    output = 1000;

  return output;
}

/*
 * Adjusts the tension in line based on error output
 * from pidControl()
 */
int pwmSignal()
{
  int tension;
  int input = pidControl();

  if(input > 50)
    tension = hover - input * .030;

  else if(input < -50)
    tension = hover - input * .065;

  else
    tension = hover;

  return tension;
}

/*
 * Switches between free flight and reel in mode each time
 * button is pressed
 */
int getSwitchMode()
{
  int manualMode = digitalRead(manualSwitchPin);
  if(manualMode != lastButtonState)
    lastDebounce = millis();

  if((millis() - lastDebounce) > debounceDelay){
    if(manualMode != buttonState){
      buttonState = manualMode;

      if(buttonState == HIGH)
        manualOn = !manualOn;
    }
  }

  lastButtonState = manualMode;
  return manualOn;
}

/*
 * Functions for motor control in different modes
 */
void freeFlight()
{
  analogWrite(motorPin, pwmSignal());
}

void reelIn()
{
  analogWrite(motorPin, pullIn);
}

void loop()
{
  static int mode = 0;
  mode = getSwitchMode(); //user pushes button to reel in

  if(mode == 0) //autonomous flight
    freeFlight();

  if(mode == 1){ //motor pulls in quad
    reelIn();
    iPart = 0; //freezes error
  }
  /*
   //Prints out info to screen
   Serial.print(" mode = ");
   Serial.print(mode);
   Serial.print("\tDuty Cycle = ");
   if(mode == 0)
     Serial.print(pwmSignal() * 100 / 255);
   else
     Serial.print(pullIn * 100 / 255);
   Serial.print(" %\tError = ");
   Serial.print(error);
   Serial.print("\tRaw = ");
   Serial.print(analogRead(analogModePin));
   Serial.print("\tFilter = ");
   Serial.println(sum);
   */
}
```

V. Acknowledgment

The author would like to acknowledge the support of the National Science Foundation, through NSF REU grant no. 1062672.

The author would like to thank the mentors who contributed to the development of this device which include: James Yang, Joseph Trovato, and Patrick Husson. Special thanks to my advisor, Dr. Daniel D. Lee and to the Sunfest director Dr. Jan Van der Spiegel.

VI. References

[1] "Unmanned Aircraft Systems (UAS) Operational Approval," FAA, Washington, DC, N 8900.207, Jan. 2013.

[2] **(**2009, June**).** Multi Autonomous Ground-robotic International Challenge 2010. Defence Science and Technology Organisation and US Army RDECOM. Endinburah, Austrailia. [Online] Available: http://www.dsto.defence.gov.au/attachments/MAGIC%202010%20final_2.pdf

[3] S. Thrun, Simultaneous Localization and Mapping, Robotics and Cognitive Approaches to Spatial Mapping Springer Tracts in Advanced Robotics, 38 (2008) 13-41.
[4] T. Wescot. (2000). *PID without a PhD*. EE Times-India [Online]. 10(9) pp, 1-7. Available: http://tec.upc.es/mp/2012/PID-without-a-PhD.pdf
[5] G. Goebel. (2012, Feb.). *Unmanned Aerial Vehicles: USA*. (1st ed.) [Online]. Available: http://www.vectorsite.net/twuav.html
[6] G. M. Hoffman, H. Huang, S. L. Washlander, C. J. Tomlin, "Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment American Institute of Aeronautics and Astronautics, Hilton Head, SC, Aug. 23, 2007.
[7] E. Ramsden, "Hall-Effect Physics," in Hall-Effect Sensors: Theory and Application, 2nd ed. Oxford, UK: Elsevier Inc., 2006, ch. 1, pp. 1-5.
[8] "First-Order Exponential Filter," in NeurOn-Line User's Guide, 1st ed. Cambridge: Gensym Corporation, Inc., 1996, ch 4.