Программа управления робота-сумоиста.

Кучин Д.Э., Рябинкин С.М., Грач Е.П., Колледж приборостроения и информационных технологий

В ЦТПО МИРЭА функционирует кружок спортивной робототехники. Самой интересной номинацией является сумо роботов. Это одна из немногих соревновательных дисциплин, в которых роботы команд-соперников в прямом смысле слова противостоят друг другу. В этом состязании участникам необходимо подготовить автономного робота, способного наиболее эффективно выталкивать робота-противника за пределы ринга. Создание робота-сумоиста - посильная задача, и именно поэтому данная номинация одна из самых популярных. Практика показывает, что любой технически слабый робот превращается в грозного соперника, если программа управления написана грамотно.

Алгоритм управления сумоиста очень прост. Найти соперника и вытолкать соперника за ринг. Получается, что робот должен уметь совершать два основных маневра – поиск и таран. В случае использования датчиков Sharp 340 с заявленной дистанцией обнаружения 40 см, необходимо сразу после старта сократить дистанцию до соперника, т.к. по правилам мини-сумо, ринг имеет диаметр 77 см. В номинации микро-сумо данный маневр не нужен. В случае использования датчиков линии, необходимо предусмотреть четвертый маневр – отъезд от края ринга.

Рассмотрим робота, у которого на борту установлено 5 датчиков. Три барьерных дальномера Sharp 340 и два датчика линии.

Первый датчик Sharp 340 - курсовой. Это главный датчик робота. Два других датчика дополнительные. Они располагаются под углом 90 градусов в разные стороны относительно продольной оси робота. Их роль - обнаружение противника и определение направления поворота робота.

Датчики Sharp 340 - барьерные цифровые. Высокий логический уровень (5В) - нет объекта в зоне обзора датчика. Низкий логический уровень (0В) - в зоне обзора находится объект, при этом отсутствует информация о дистанции до объекта, о его размерах.

Датчики линии используются как правило аналоговые. Формально, у них 1024 возможных значения (при использовании стандартных настроек 10-разрядного АЦП микропроцесора ATmega 328), однако на практике робот может располагаться либо на черной части соревновательного ринга (что соответствует показанию датчика от 0 до 100), либо на белой (и тогда преобразованный сигнал датчика будет в диапазоне от 950 до 1023 единиц АЦП). Это означает, что показания датчика линии сводятся к двум возможным состояниям: робот на черной части ринга, робот на белой части ринга.

Таким образом, у робота будет всего 2^5 возможных состояния по показаниям датчиков. При этом сразу можно исключить тот набор показаний датчиков, который невозможен в принципе. Например, соперник не может находится одновременно слева и справа. Это состояние датчиков может говорить только о том, что один из датчиков неисправен.

Программу управления для Arduino-совместимых микроконтроллеров удобно писать в Arduino IDE. Стандартная конструкция программы в Ardiuno IDE предусматривает два главных метода программы. Метод *setup()* - метод, выполняющийся один единственный раз при подаче питания на микроконтроллер. Метод *loop()* - является аналогом бесконечного цикла. Этот метод будет выполняться постоянно, пока на микроконтроллер подается питание.

В методе *setup()* устанавливаются состояние ножек (пинов) микроконтроллера. Состояние INPUT соответствует входу. Все пины микроконтроллера, к которым подключены датчики, необходимо перевести в состояние INPUT. Это делается один раз, при включении микроконтроллера, то есть в методе setup(). Состояние OUTPUT соответствует выходу. Пины микроконтроллера, к которым подключены управляющие входы драйвера двигателей, необходимо переключить в состояние OUTPUT. Это делается так же всего один раз, при включении микроконтроллера. После установки этих выходов в состояние OUTPUT на них необходимо установить сигналы, соответствующие полной остановке робота.

```
void setup()
{
  pinMode (8, INPUT); //центральный
  pinMode (4, INPUT);//правый датчик
  pinMode (12, INPUT); //левый датчик
  pinMode(A0,INPUT);//датчик линии левый
  pinMode(A1,INPUT);//датчик линии правый

  pinMode (2, INPUT); //стартовый ИК-модуль

  pinMode (6, OUTPUT);//мотор лево назад
  pinMode (9, OUTPUT); //мотор лево вперед
  pinMode (3, OUTPUT);//мотор правый назад
  pinMode (5, OUTPUT);//мотор право вперед

  digitalWrite(6, LOW);
  digitalWrite(9, LOW);
  digitalWrite(3, LOW);
  digitalWrite(5, LOW);


}//void setup()
void loop() {


}//void loop()
```

Рис.1. Главный метод программы управления робота в Arduino IDE. Устанавливаются режимы пинов микроконтроллера, моторы робота останавливаются.

Регламент соревнований предусматривает наличие на роботе специального стартового модуля. Стартовый модуль в некотором роде является шестым датчиком робота, и его показания должны иметь наивысший приоритет. Пин микроконтроллера, к которому подключен стартовый модуль, должен быть так же переведен в состояние INPUT.

Стартовый модуль выдает логический ноль в режиме "Power ON" и "Stopped", и логическую единицу в режиме "Started". После подачи питания на микроконтроллер (включение робота) необходимо дождаться переключения стартового модуля в режим "Started", то есть пока на соответствующем входе сохраняется логический ноль, не происходит ничего.

После переключения входа на логическую единицу выполнение цикла прервется, начнется выполнение команд следующих после него. Перед переключением робота в режим поиска, необходимо сократить расстояние до соперника. Заявленная дальность действия датчиков Sharp 340 - 40 см, в то время как диаметр ринга в номинации "мини-сумо" - 77 см. Выходы микроконтроллера, соответствующие движению вперед переключаются в высокий логический уровень. После этого выполнится команда *delay(150),* которая является аналогом паузы. То есть в течение 150 мс микроконтроллер не будет делать ничего. При этом состояние выходов, соответствующих движению робота вперед останется неизменным. Иначе говоря, 150 мс робот будет двигаться вперед (см. рис.2). На этом выполнение команд в методе *setup()* закончится, начнется бесконечный вызов метода *loop().*

```
void setup()
{
  pinMode (8, INPUT); //центральный
  pinMode (4, INPUT);//правый датчик
  pinMode (12, INPUT); //левый датчик
  pinMode(A0,INPUT);//датчик линии левый
  pinMode(A1,INPUT);//датчик линии правый

  pinMode (2, INPUT); //стартовый ИК-модуль

  pinMode (6, OUTPUT);//мотор лево назад
  pinMode (9, OUTPUT); //мотор лево вперед
  pinMode (3, OUTPUT);//мотор правый назад
  pinMode (5, OUTPUT);//мотор право вперед

  digitalWrite(6, LOW);
  digitalWrite(9, LOW);
  digitalWrite(3, LOW);
  digitalWrite(5, LOW);

  while (!digitalRead(2))continue; //ожидаем старта с судейского пульта

  digitalWrite(9, HIGH); //мотор правый крутится вперед
  digitalWrite(5, HIGH); //мотор левый крутится вперед
  delay(150);// движение вперед в течение 150 мс

}//void setup()
void loop() {


}//void loop()
```

Рис.2. Циклом прописано ожидание старта с судейского пульта. Три следующих команды обеспечивают выезд робота на центр ринга.

Каждое выполнение метода *loop()* (каждый шаг бесконечного цикла) должен начинаться с опроса датчиков. Показания датчиков сохраним в переменные с "говорящими" именами. При этом следует помнить, что показания датчиков инвертированы с точки зрения здравого смысла. Логическая единица - дальномер не видит ничего. Логический ноль - дальномер что-то видит. 0 единиц АЦП - робот на абсолютно белой поверхности, 1023 единицы АЦП - робот на абсолютно черной поверхности. Для удобства рекомендуем "перевернуть" показания датчиков операцией логического инвертирования.

Показания датчиков линии так же сводятся к бинарному "черное-белое" с помощью условного оператора *if*. Границей между белым и черным является 512 единиц АЦП (см. рис.3).

```
}//void setup()
void loop() {
  int center = !digitalRead(8);//центральный датчик. показания инвертированы.
  int left = !digitalRead(12);//левый датчик. показания инвертированы.
  int right = !digitalRead(4);//правый датчик. показания инвертированы.
  int start_IK = digitalRead(2);//стартовый ИК-модуль

  int line_left, line_right;
    if(analogRead(A0)<512)
      line_left=0;
    else
      line_left = 1;
    if(analogRead(A1)<512)
      line_right=0;
    else
      line_right = 1;

}//void loop()
```

Рис. 3. Главный метод программы. Опрос датчиков. Обращаем внимание, что инициализация переменных line_left и line_right должна проводиться вне условного оператора if.

Далее необходима проверка на состояние стартового модуля. Если стартовый модуль переключен в режим "Stopped", то в соответствующей переменной будет записан логический ноль. В этом случае пины микроконтроллера, к которым подключены управляющие входы драйвера двигателей устанавливаются в состояние, соответствующее остановке робота. Далее начнется выполнение бесконечного цикла, из которого робота можно вывести, только перезагрузкой микроконтроллера.

Следующий оператор *while* отвечает за маневр тарана. Если центральный датчик обнаружил объект, то в соответствующей переменной будет записана логическая единица. В этом случае пины, к которым подключены входы драйвера двигателей, переключаются в состояние, соответствующее движению вперед с максимальной скоростью. И не забываем на каждом шаге этого цикла опрашивать центральный датчик (см. рис.4).

```
if(!start_IK){ //обеспечит остановку робота по сигналу с судейского пульта
    digitalWrite(6, LOW);
    digitalWrite(9, LOW);
    digitalWrite(3, LOW);
    digitalWrite(5, LOW);
    while(true)continue;
  }//if(!start_IK)
 while(center){  //обеспечит маневр тарана
    digitalWrite(6, LOW);
    digitalWrite(9, HIGH);
    digitalWrite(3, LOW);
    digitalWrite(5, HIGH);
    center=!digitalRead(8);

  } //while(center)
```

Рис.4. Прописана остановка по сигналу с пульта, прописан маневр тарана по сигналу центрального датчика.

Следующие по приоритету - датчики линии. Если хотя бы один из них в состоянии, соответствующем наезду на белый край ринга, то в этом случае робот движется назад 80 мс, а затем совершает танковый разворот в течение 100 мс (см. рис. 5).

```
if(!line_left || !line_right){
      digitalWrite(9, LOW);//переключаем пины моторов в состояние,
      digitalWrite(6, HIGH);//соответствующее движению назад
      digitalWrite(5, LOW);
      digitalWrite(3, HIGH);
      delay(80);// отъезжаем назад 80 мс.
      digitalWrite(6, LOW);//переключаем пины моторов в состояние,
      digitalWrite(9, HIGH);//соответствующее танковому развороту
      digitalWrite(5, LOW);
      digitalWrite(3, HIGH);
      delay(100);//совершаем танковый разворот 100 мс.
  }//if(!line_left || !line_right)
```

Рис. 5. Прописан маневр отъезда от края ринга. Следует помнить, что мало просто переключить пины электромоторов, необходимо еще это состояние сохранять в течение какого-то времени.

Следующие два условных оператора *if* напрямую не влияют на состояние пинов драйвера двигателей. Они лишь устанавливают значение двум переменным, отвечающие за скорости вращения двигателей. Во-первых, такая конструкция операторов позволяет исключить ситуацию, описанную в начале статьи, когда по каким-то причинам и левый и правый датчик заметили противника. Последовательное выполнение операторов *if* приведет к тому, что в этой ситуации достоверным считается показание левого датчика. Во-вторых, отсутствие изменения этих двух переменных в других местах программы позволит роботу при совершении поворота к противнику преодолеть 90-градусную "мертвую" зону, когда боковой датчик УЖЕ не видит противника, а центральный ЕЩЕ не видит.

Далее следует простая конструкция if-else, которая установит направление и скорость вращения электродвигателей робота (см. рис.6.).

```
if(right){
  speed_left=80;
  speed_right=-80;
  }//if(right
if(left){
  speed_left=-80;
  speed_right=80;
  }//if(left)

  if(speed_left<0 && speed_right>0){
    analogWrite(6, 80);
    digitalWrite(9, LOW);
    digitalWrite(3, LOW);
    analogWrite(5, 80);

  }//if(speed_left<0 && speed_right>0)
  else{
    digitalWrite(6, LOW);
    analogWrite(9, 80);
    analogWrite(3, 80);
    digitalWrite(5, LOW);

  }//else
```

Рис. 6. Маневр поиска и поворота к сопернику. Переменные speed_left и speed_right должны быть объявлены как глобальные, иначе режим поиска будет работать некорректно. Так же обращаем внимание, что при использовании ШИМ, необходимо применять функцию analogWrite().

Данный алгоритм управления отлично подходит для простых роботов с малым количеством датчиков. Роботы созданные в ЦТПО МИРЭА в ЦАО под управлением такого алгоритма побеждали и занимали призовые места на многих турнирах, в том числе на престижном Робофесте-2016.