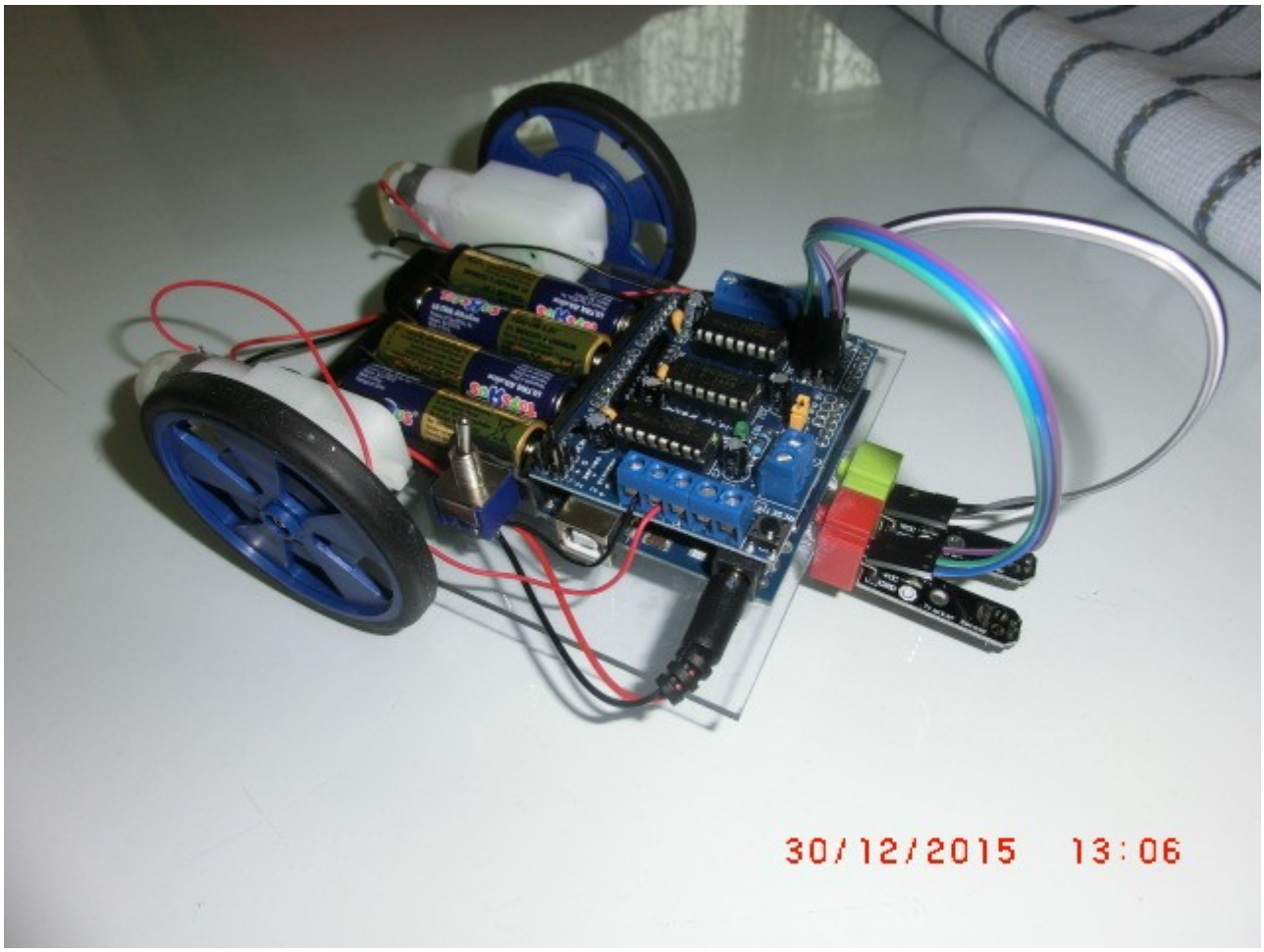
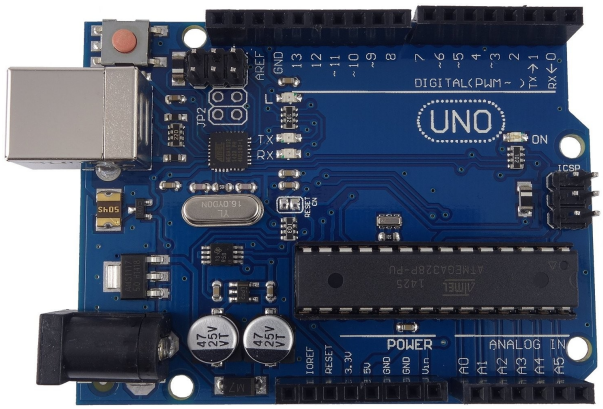
Arduino Line Follower Roboter

mail@AndreBetz.de

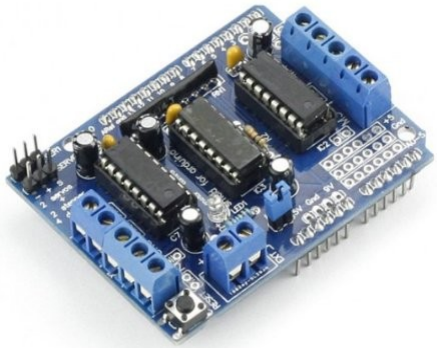


1. Bauteile

Arduino UNO R3



SaintSmart LD293D Motor Driver Shield



2x TCRT5000 Line Tracking Sensor

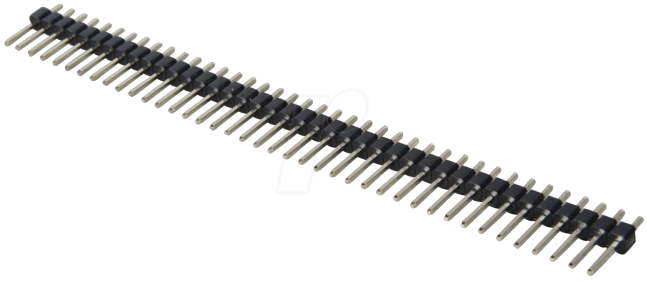
Switch

Batteriehalter 4xAA Mignon

2x 3-6V GM9 Getriebemotor

2x 67mm Rad für GM9

4x Pins Male Leiste



9V Batterieclip auf 2,1mm x 5,5mm Stecker

4x Jumper Wire female-female

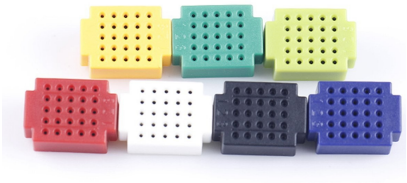
USB Kabel für Arduino UNO R3 Board

Doppelseitiges Klebeband

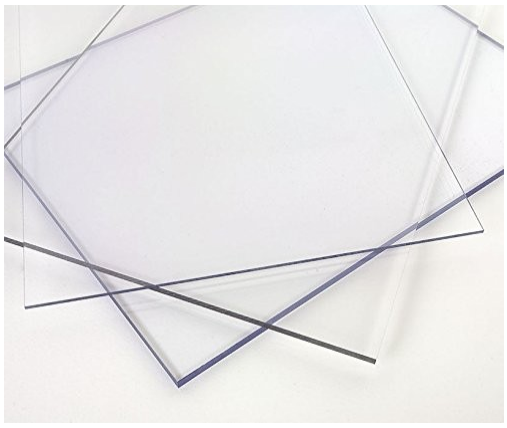
Ball Caster 3/8

- Sekundenkleber

- Mini Breadbord



- Polycarbonat Platte



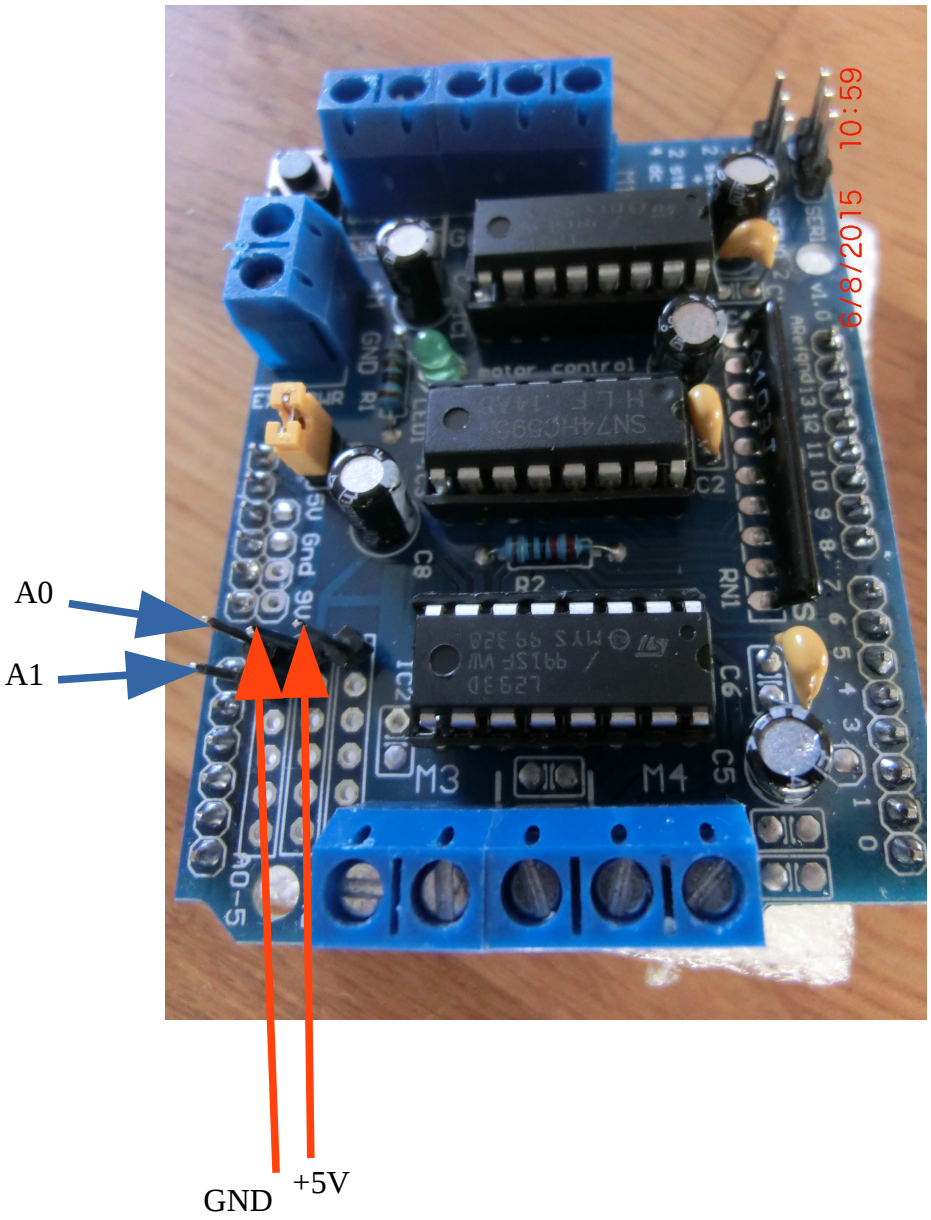
- 1cm breites schwarzes klebendes Isolierband

2. Werkzeug

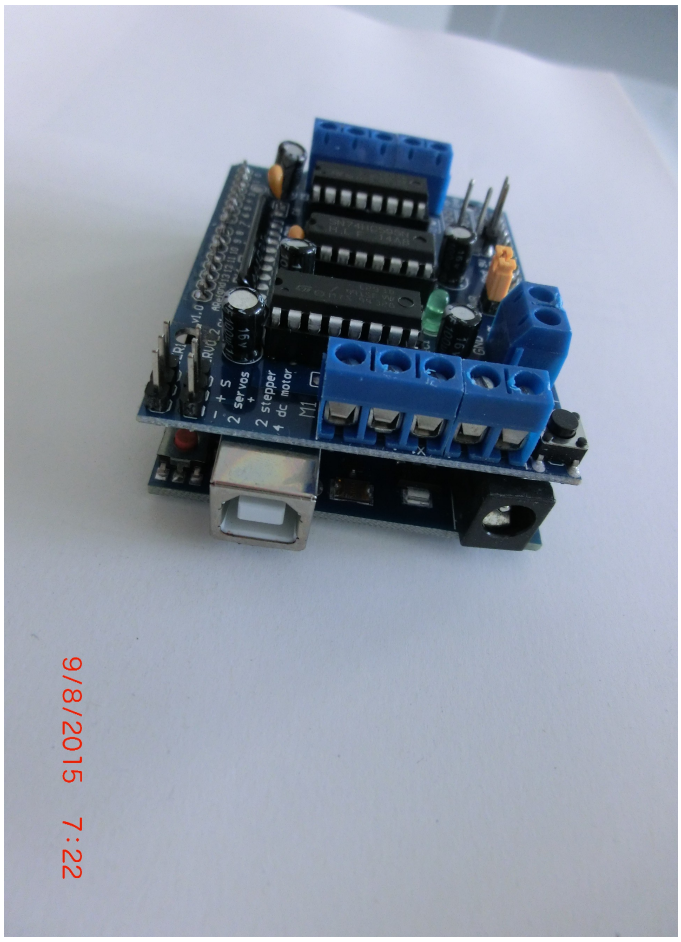
- Lötkolben
- Lötzinn
- Schraubenzieher

3. Zusammenbau

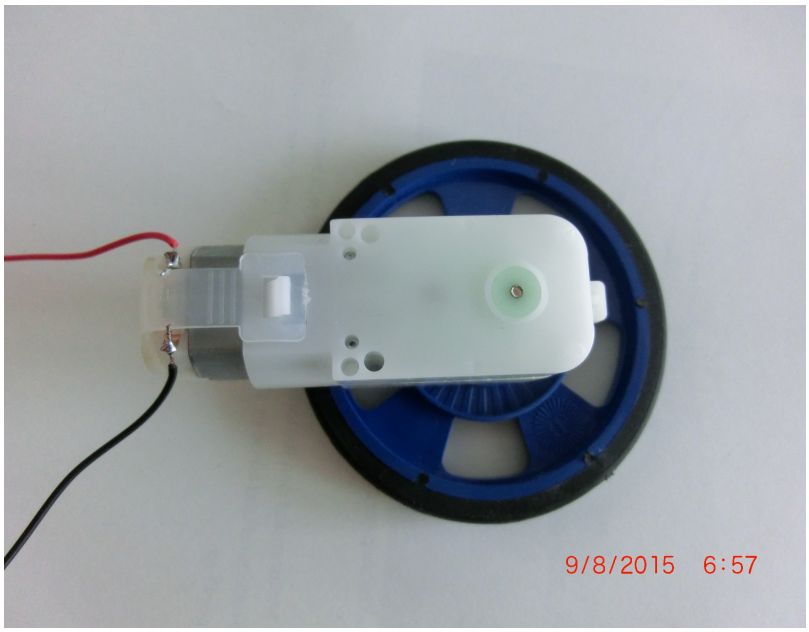
Anlöten der 6 Pins auf das Motorshield zum Anschluss der Line Tracking Sensoren. Diese werden am analog Eingang A0 und A1 ausgelesen



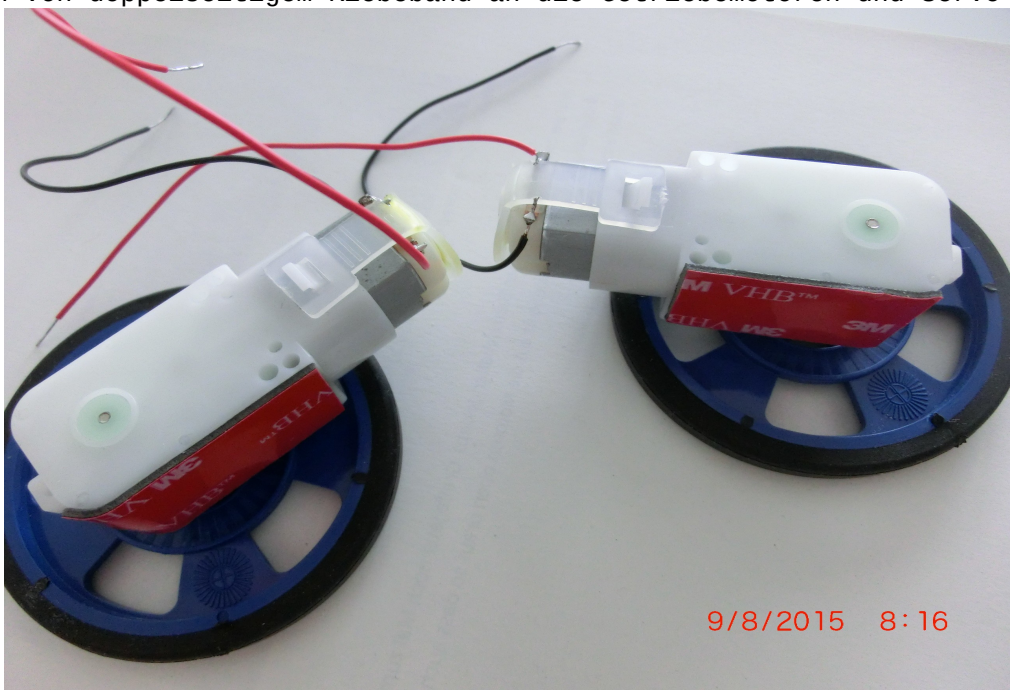
Aufstecken des Motor Shieldes auf das Arduino Board



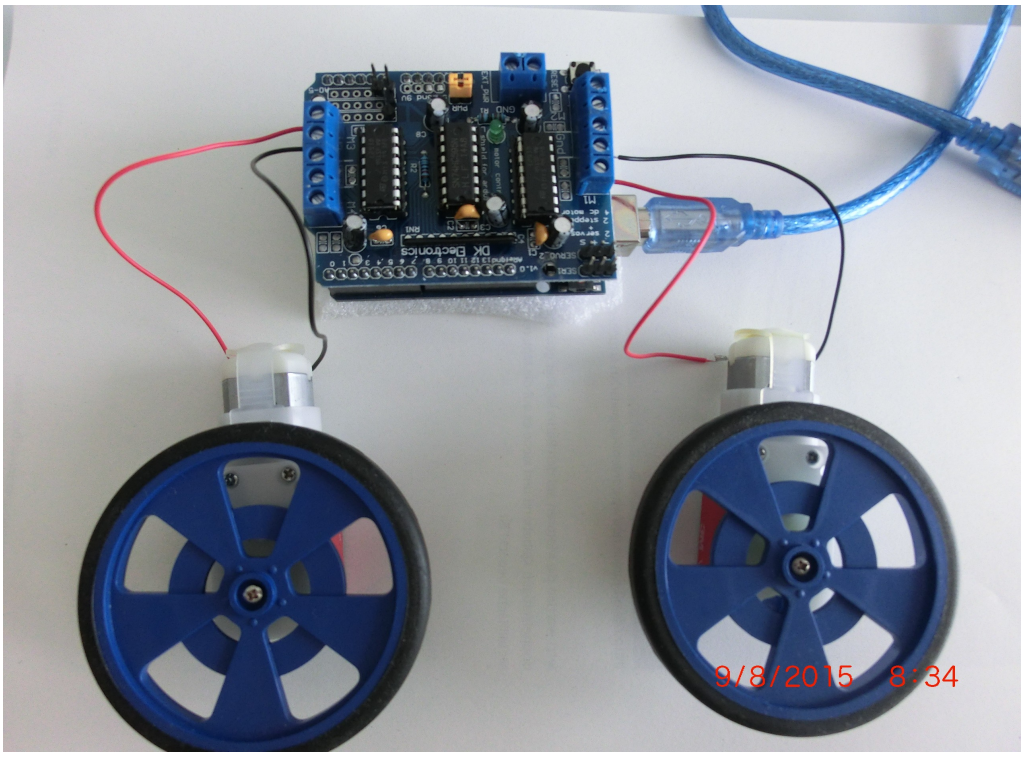
Zusammenbau der Räder an die Getriebemotoren GM9 und anlöten der Drähte



Anbringen von doppelseitigem Klebeband an die Getriebemotoren und Servo

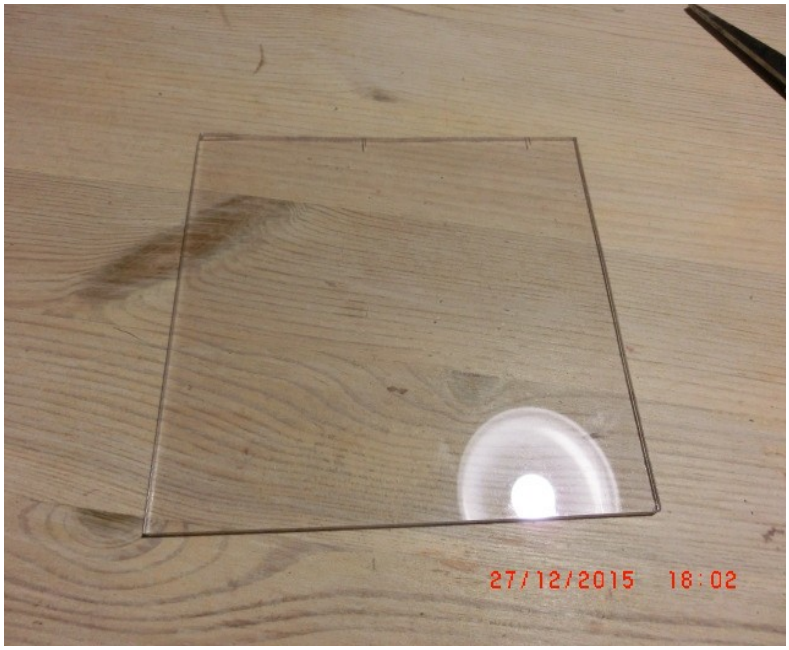


Verbinden der Getriebemotoren mit dem Motor Shield an M1 und M3

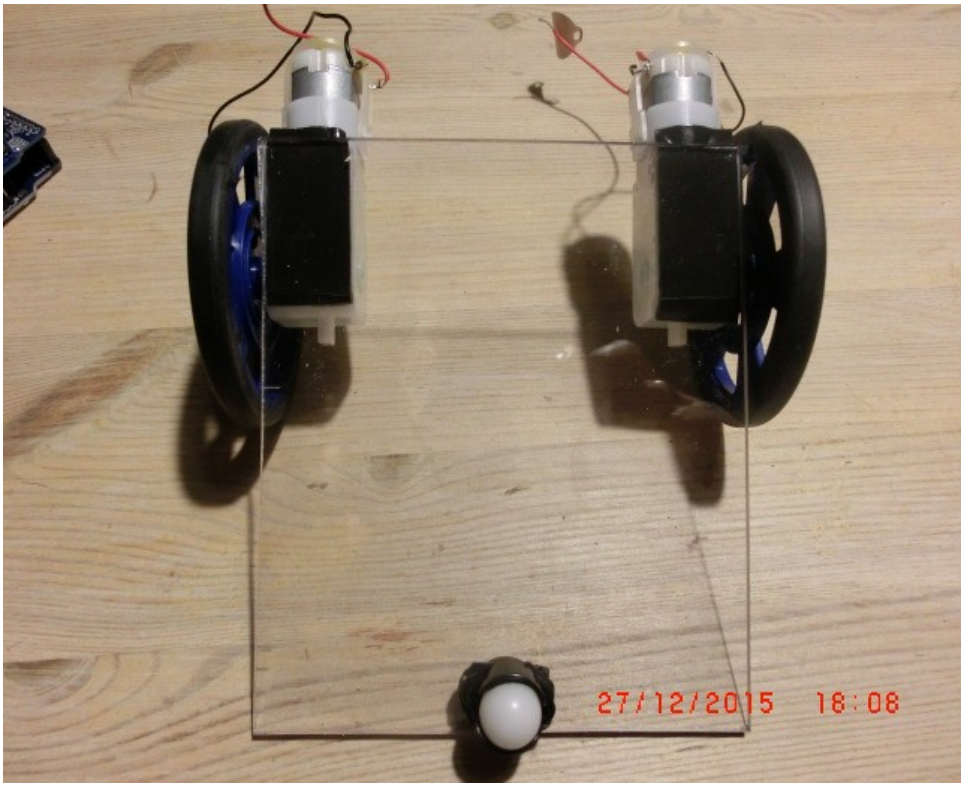


Später ist darauf zu achten, dass die Räder sich beim Fahren in die gleiche Richtung drehen. Sollte das nicht der Fall sein, so einfach die Poolung verauschen.

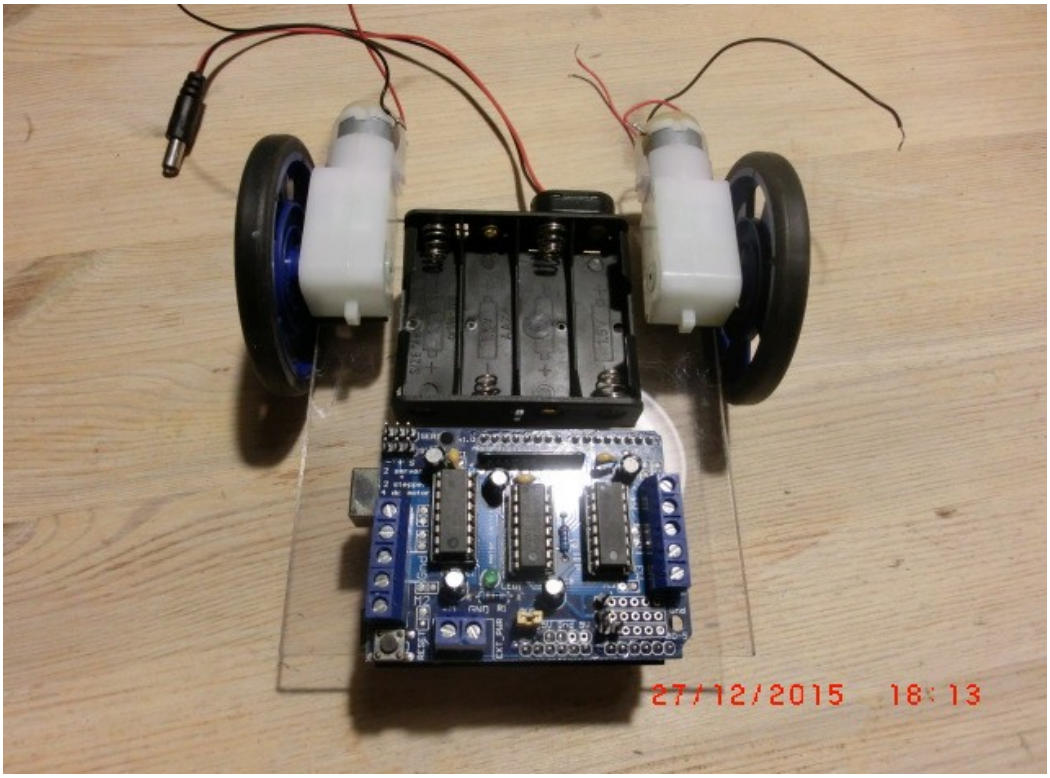
Zurechtschneiden der Polycarbonatplatte auf die Grösse 12cm x 13cm



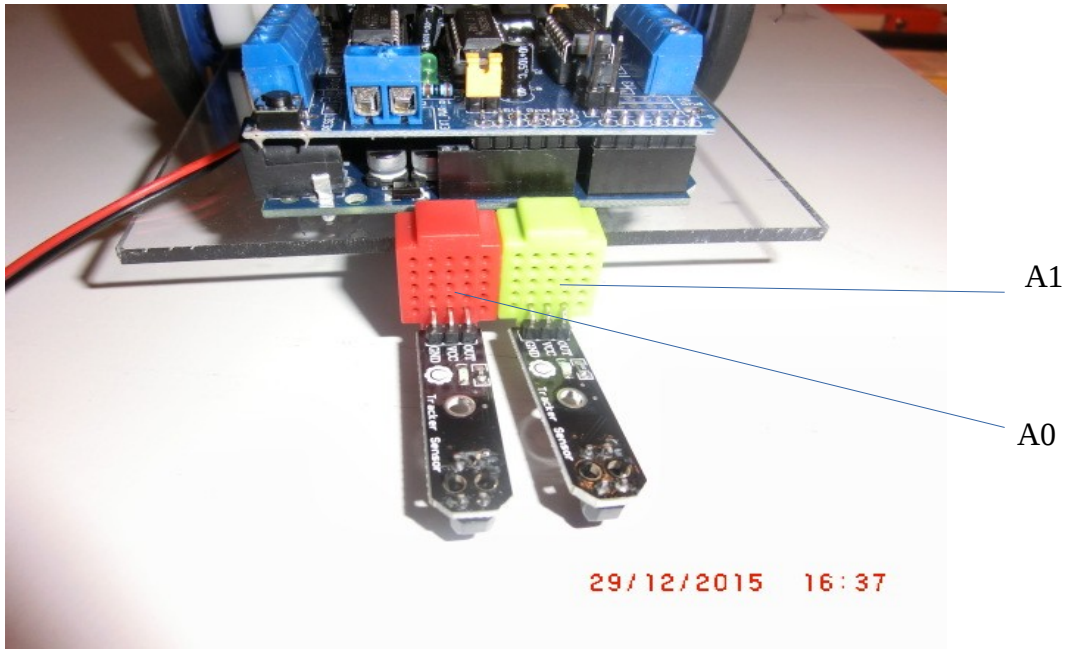
Aufkleben der Getriebemotoren parallel zu einander und des Ball Casters



Aufkleben der Batteriehalterung und des Arduino Bordes mit Motorsteuerung



Anbringen der mini Breadbords mit den zwei Linienverfolgern. Die Breadbords sind mit Sekundenkleber auf die Polycarbonatplatte aufgeklebt. Zusaätzlich sind die Pins auf dem Breadbord auch angeklebt, damit diese nichtwackeln. Der Sensor sollte ca 1cm Abstand vom Boden haben.



Der Sensor hat 3 Anschlüsse: GND, VCC, Out.
GND an GND und VCC an +5V auf der Platine anschliessen. Der linke Out wird an A0 und der Rechte an A1 des analogen Einganges des Arduino auf der Motorsteuerplatine angeschlossen.

4. Software

4.1 benötigte Arduino Bibliotheken:

AFMotor: Arduino Motor Shield Library von AdaFruit
https://github.com/adafruit/Adafruit-Motor-Shield-library

4.2 Arduino Code:

```
#include <stdarg.h>
#include <AFMotor.h>

#define RIGHT_MOTOR   1
#define LEFT_MOTOR    3
#define RIGHT_SENSOR  A0
#define LEFT_SENSOR   A1

class MyMotor
{
  public:
    MyMotor(int number) :
      mMotor(number),
      mCurrentSpeed(0)
    {
    }
    void setSpeed(int speed)
    {
      mCurrentSpeed = speed;
      if (speed >= 0)
      {
        mMotor.setSpeed(speed);
        mMotor.run(FORWARD);
      }
      else
      {
        mMotor.setSpeed(-speed);
        mMotor.run(BACKWARD);
      }
    }
    void stop()
    {
      setSpeed(0);
    }
    void forwardFast()
    {
      if ( 0 == mCurrentSpeed )
        setSpeed(200);
    }
    void forwardSlow()
    {
      if ( 0 == mCurrentSpeed )
        setSpeed(170);
    }
    void backwardSlow()
    {
      if ( 0 == mCurrentSpeed )
        setSpeed(-170);
    }
  private:
    AF_DCMotor mMotor;
```

```
    int mCurrentSpeed;
};

class Sensor
{
  public:
    Sensor(int analogPin) :
      mPin(analogPin),
      mThreshold(500)
    {
    }
    void init()
    {
      pinMode(mPin, INPUT);
    }
    boolean isDark()
    {
      int sensorValue = analogRead(mPin);
      if ( sensorValue < mThreshold )
        return true;
      else
        return false;
    }
  private:
    int mPin;
    int mThreshold;
};

class LineFollowRobot
{
  public:
    LineFollowRobot() :
      mLeftMotor(LEFT_MOTOR),
      mRightMotor(RIGHT_MOTOR),
      mLeftSensor(LEFT_SENSOR),
      mRightSensor(RIGHT_SENSOR)
    {
    }
    void init()
    {
      mLeftSensor.init();
      mRightSensor.init();
      state = stateStop;
    }
    void run()
    {
      boolean leftBlack = mLeftSensor.isDark();
      boolean rightBlack = mRightSensor.isDark();

      if ( stateMoving == state )
      {
        if ( rightBlack ||
             leftBlack )
        {
          stop();
        }
        else
        {
          move();
        }
      }
      else if ( stateStop == state )
      {
        if ( leftBlack )
```

```
      {
        turnLeft();
      }
      else if ( rightBlack )
      {
        turnRight();
      }
      else
      {
        move();
      }
    }
    else if ( stateTurnLeft == state )
    {
      if ( false == leftBlack )
      {
        stop();
        state = stateMoving;
        }
    }
    else if ( stateTurnRight == state )
    {
      if ( false == rightBlack )
      {
        stop();
        state = stateMoving;
        }
    }
  }
private:
  void move()
  {
    mLeftMotor.forwardFast();
    mRightMotor.forwardFast();
    state = stateMoving;
  }
  void stop()
  {
    mLeftMotor.stop();
    mRightMotor.stop();
    state = stateStop;
  }
  void turnLeft()
  {
    //mLeftMotor.backwardSlow();
    mRightMotor.forwardSlow();
    state = stateTurnLeft;
  }
  void turnRight()
  {
    //mRightMotor.backwardSlow();
    mLeftMotor.forwardSlow();
    state = stateTurnRight;
  }

  enum state_type {
    stateMoving,
    stateStop,
    stateTurnLeft,
    stateTurnRight};
  state_type state;

  MyMotor mLeftMotor;
  MyMotor mRightMotor;
```

```
    Sensor  mLeftSensor;
    Sensor  mRightSensor;
};

LineFollowRobot robot;

void setup() {
  Serial.begin(9600);
  robot.init();
}

void loop() {
  robot.run();
}
```